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)

The goal of modeling

Collecting a priori knowledge

A priori model

Experiment design

Observations, determining features, parameters

Model validation

Final model

Correction

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
Excitation

- **Multisine**
  \[ u(k) = \sum_{k=1}^{K} U_k \cos \left( 2\pi \frac{k}{N} f_{\text{max}} + \varphi(k) \right) \]
  - where \( f_{\text{max}} \) is the maximum frequency of the excitation signal,
  \( K \) is the number of frequency components

- **Crest factor**
  \[ CF = \frac{\max(u(t))}{u_{\text{rms}}(t)} \]
  minimizing \( CF \) with the selection of \( \varphi \) 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
    - variant model
  - frequency dependent

\[ U(f) = Z(f)I(f) \quad Z(f) = \frac{U(f)}{I(f)} \quad Z(f) = \frac{R}{j2\pi fRC+1} \]

\[ U = RI \quad U = R(I)I \]
Modeling (example)

- Resistor modeling
  - nonparametric model

![Graph showing linear and nonlinear behavior in DC and AC conditions.](image)
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

![Diagram of a duct with noise sources and error signal](image-url)
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

• Output of a neural model

![Diagram of time series output](image-url)
References and further readings

Identification (linear systems)

• Parametric identification (parameter estimation)
  - LS estimation
  - ML estimation
  - Bayes estimation

• Nonparametric identification
  - Transient analysis
  - Correlation analysis
  - Frequency analysis
Parametric identification

\[ y = f(u, n) \]

\[ y_M = f_M(u, \theta) \]

\[ C(y, y_M) \]

System function

Model

Criterion function

Parameter adjustment algorithm
Parametric identification

- Parameter estimation
  - linear system

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

\[ U = \begin{bmatrix} u(1)^T \\ \vdots \\ u(N)^T \end{bmatrix} \quad y = U\Theta + n \]

\[ y^T = y_N^T = [y(1) \quad \cdots \quad y(N)] \]

- linear-in-the parameter model

\[ y_M(i) = u(i)^T \hat{\Theta} = \sum_{j} u_j(i) \hat{\Theta}_j \quad y_M = U\hat{\Theta} \]

- criterion (loss) function

\[ \varepsilon(\Theta) = y - y_M(\Theta) \quad V(\hat{\Theta}) = V(\varepsilon(\Theta)) = V(y - y_M) = V(y - y_M(\Theta)) \]
Parametric identification

- LS estimation

quadratic loss function

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

\[ \frac{1}{2} \sum_{i=1}^{N} (y(i) - u(i)^T \Theta)(y(i) - u(i)^T \Theta) = \frac{1}{2} (y_N - U\Theta)^T (y_N - U\Theta) \]

LS estimate

\[ \hat{\Theta}_{LS} = \arg \min_{\Theta} V(\Theta) \quad \frac{\partial V(\Theta)}{\partial \Theta} = 0 \]

\[ \hat{\Theta}_{LS} = (U_N^T U_N)^{-1} U_N^T y_N \]
Parametric identification

- Weighted LS estimation
  - weighted quadratic loss function
  \[
  V(\Theta) = \frac{1}{2} \sum_{i=1}^{N} \varepsilon(t)^2 = \frac{1}{2} \sum_{i,k=1}^{N} (y(i) - u(i)^T \Theta)q_{ik}(y(k) - u(k)^T \Theta) = \frac{1}{2} (y_N - U\hat{\Theta})^T Q(y_N - U\hat{\Theta})
  \]
  weighted LS estimate
  \[
  \hat{\Theta}_{WLS} = (U_N^T Q U_N)^{-1} U_N^T Q y_N
  \]
  - Gauss-Markov estimate (*BLUE=best linear unbiased estimate*)
  \[
  \begin{align*}
  E\{n\} &= 0 & \text{cov}[n] &= \Sigma & Q &= \Sigma^{-1} \\
  \hat{\Theta}_{WLS} &= (U_N^T \Sigma^{-1} U_N)^{-1} U_N^T \Sigma^{-1} y_N
  \end{align*}
  \]
Parametric identification

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

\[
f(y|\hat{\Theta}_1) \quad f(y|\hat{\Theta}_{ML}) \quad \ldots \quad f(y|\hat{\Theta}_k)
\]

- likelihood function, log likelihood function

\[
f(y_N|\hat{\Theta}) \quad \log f(y_N|\hat{\Theta})
\]

- maximum likelihood estimate

\[
\hat{\Theta}_{ML} = \arg \max_{\Theta} f(y_N|\hat{\Theta}) \quad \frac{\partial}{\partial \hat{\Theta}} \log f(y_N|\hat{\Theta}) = 0
\]
Parametric identification

- Properties of ML estimates
  - consistency
    \[ \lim_{N \to \infty} P \left( \left| \hat{\Theta}_{ML(N)} - \Theta \right| > \varepsilon \right) = 0 \quad \text{for any} \quad \varepsilon > 0 \]
  - asymptotic normality
    \[ \hat{\Theta}_{ML(N)} \quad \text{converges to a normal random variable as} \quad N \to \infty \]
  - asymptotic efficiency: the variance reaches Cramer-Rao lower bound
    \[ \lim_{N \to \infty} \text{var}(\hat{\Theta}_{ML(N)} - \Theta) = \left( E \left( \frac{\partial^2 \ln f(y|\Theta)}{\partial \Theta^2} \right) \right)^{-1} \]
  - Gauss-Markov if \( f(y_N|\hat{\Theta}) \) Gaussian
Parametric identification

- Bayes estimation
  - the parameter $\Theta$ is a random variable with known pdf

  $$V_B(\hat{\Theta}) = \int C(\hat{\Theta}|\Theta)f(\Theta|y)d\Theta$$

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

- Bayes estimation with different cost functions

- median \[ C(\hat{\Theta} | \Theta) = |\hat{\Theta} - \Theta| \]

- MAP \[ C(\hat{\Theta} | \Theta) = \begin{cases} 
\text{Const} & \text{if } |\hat{\Theta} - \Theta| \leq \Delta \\
0 & \text{otherwise}
\end{cases} \]

- mean \[ C(\hat{\Theta} | \Theta) = \|\hat{\Theta} - \Theta\|^2 \]
Parametric identification

- Recursive estimations

  - \( \hat{\Theta}(k) \) is estimated from \( \{y(i)\}_{i=1}^{k-1} \)

  - \( y(k) \) is predicted as \( y_M(k) = u(k)^T \hat{\Theta} \)

  - the error \( e(k) = y(k) - y_M(k) \) is determined

  - update the estimate \( \hat{\Theta}(k+1) \) from \( \hat{\Theta}(k) \) and \( e(k) \)
Parametric identification

- Recursive estimations
  - least mean square \( LMS \)

\[
\hat{\Theta}(k+1) = \hat{\Theta}(k) + \mu(k)\varepsilon(k)u(k)
\]

- the simplest gradient-based iterative algorithm

- it has important role in neural network training
Parametric identification

- Recursive estimations
  
  - recursive least square \textit{RLS}

\[
\Theta(k+1) = \Theta(k) + K(k+1)e(k)
\]

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

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

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

\( K(k) \) changes the search direction from instantenous gradient direction
Parametric identification

- Recursive estimations

  - recursive Bayes a posteriori df \( f(\Theta | y) \)

\[
f(\Theta | y_1) = \frac{f(y_1 | \Theta) f(\Theta)}{\int_{-\infty}^{+\infty} f(y_1 | \Theta) f(\Theta) d\Theta}
\]

\[
f(\Theta | y_1, y_2) = \frac{f(y_1, y_2 | \Theta) f(\Theta)}{\int_{-\infty}^{+\infty} f(y_1, y_2 | \Theta) f(\Theta) d\Theta}
\]

\[
f(\Theta | y_1, y_2, \ldots, y_k) = \frac{f(y_k | y_1, y_2, \ldots, y_{k-1}, \Theta) f(y_1, y_2, \ldots, y_{k-1}, \Theta)}{\int_{-\infty}^{+\infty} f(y_k | y_1, y_2, \ldots, y_{k-1}, \Theta) f(y_1, y_2, \ldots, y_{k-1}, \Theta) d\Theta}
\]
Parametric identification

• Parameter estimation
  – Least square
  – Maximum Likelihood
  – Bayes

  conditional probability density $f(Y_N | \hat{\Theta})$

  a priori probability density $f(\Theta)$

  conditional probability density $f(Y_N | \Theta)$

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

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

where \( f_{\text{max}} \) is the maximum frequency of the excitation signal

\( K \) is the number of frequency components

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

minimizing \( \text{CF} \) with the selection of \( \phi \) 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


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, 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**

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

\[ \Theta = [a_1 a_2 \ldots a_N]^T \]

Parameter vector \[ \Theta = [a_1 a_2 \ldots a_N, b_1 b_2 \ldots b_P, c_1 c_2 \ldots c_L, d_1 d_2 \ldots d_K]^T \]
Black-box identification

- Non-linear dynamic model structures

**NFIR**

\[ y_M(k) = f(u(k-1), u(k-2), \ldots, u(k-N)) \]

**NARX**

\[ y_M(k) = f(u(k-1), \ldots, u(k-N), y(k-1), \ldots, y(k-P)) \]

**NOE**

\[ y_M(k) = f(u(k-1), \ldots, u(k-N), y_M(k-1), \ldots, y_M(k-P)) \]

**NARMAX**

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

**NBJ**

\[ y_M(k) = f[u(k-1), \ldots, u(k-N), y(k-1), \ldots, y(k-P), \varepsilon(k-1), \ldots, \varepsilon(k-L), \varepsilon_u(k-1), \ldots, \varepsilon_u(k-K)] \]
Black-box identification

• How to choose nonlinear mapping?

\[ y_M(k) = f(\Theta, \varphi(k)) \]

- linear-in-the-parameter models

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

- nonlinear-in-the-parameters

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

  \[ \varepsilon(k) = y(k) - y_M(k) \]

  - 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:

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

\[
r_{\varepsilon \varepsilon} = \frac{1}{\hat{C}_{\varepsilon \varepsilon}(0)} \left( \hat{C}_{\varepsilon \varepsilon}(1) \ldots \hat{C}_{\varepsilon \varepsilon}(m) \right)^T
\]

\[
\sqrt{N} r_{\varepsilon \varepsilon} \xrightarrow{\text{dist}} \mathcal{N}(0, \mathbf{I})
\]
**Black-box identification**

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

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

\[
r_{ue}(m) = \frac{1}{\sqrt{\hat{C}_{ue}(0)}} \left( \hat{C}_{ue}(\tau + 1) \ldots \hat{C}_{ue}(\tau + m) \right)^T
\]

\[
\sqrt{N} r_{ue} \xrightarrow{\text{dist}} N(0, \hat{R}_{uu})
\]

\[
\hat{R}_{uu} = \frac{1}{N - m} \sum_{k=m+1}^{N} \begin{bmatrix} u_{k-1} \\ \vdots \\ u_{k-m} \end{bmatrix} \begin{bmatrix} u_{k-1} \\ \vdots \\ u_{k-m} \end{bmatrix}^T
\]
Black-box identification

- residual test

![Auto correlation function of prediction error](chart1)

![Cross correlation function of past input and prediction error](chart2)
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:**
  \[
  \text{AIC}(\Theta) = (-2) \log(\text{max likelihood}) + 2p
  \]
  \[
  \text{AIC}(p) = (-2)\log L(\hat{\Theta}_N) + 2p
  \]

- **NIC network information criterion**
  extension of AIC for neural networks

- **MDL**
  \[
  \text{MDL}(p) = (-2)\log L(\hat{\Theta}_N) + \frac{p}{2} \log N + \frac{p}{2} \log \|\hat{\Theta}_N\|_M
  \]

\[p = \text{number of parameters}\]
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

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

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

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

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

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
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 successfully 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
References and further readings

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

\[ x_0 = 1 \rightarrow w_0 \]
\[ x_1 \rightarrow w_1 \]
\[ x_2 \rightarrow w_2 \]
\[ \ldots \rightarrow w_N \]

\[ s = \mathbf{w}^T \mathbf{x} \]

\[ y = f(s) \]

**Activation functions**

\[ y(s) = \begin{cases} 
+1 & s > 0 \\
-1 & s \leq 0 
\end{cases} \]

\[ a.) \]

\[ y(s) = \begin{cases} 
+1 & s > 1 \\
0.5 & -1 \leq s \leq 1 \\
-1 & s < -1 
\end{cases} \]

\[ b.) \]

\[ y(s) = \frac{1 - e^{-Ks}}{1 + e^{-Ks}} \quad ; \quad K > 0 \]

\[ c.) \]
Elementary neuron (2)

• Neuron with basis function

\[ y = \sum_i w_i g_i(x) \]

Basis functions \( g_i(x) = g\|x - c_i\| \) 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

\[ y = \text{sgn}(s) \]

\[ s = w^T x \]

\[ x_0 = 1 \]

\[ x_1 \]

\[ x_2 \]

\[ x_N \]

\[ w_0 \]

\[ w_1 \]

\[ w_2 \]

\[ w_N \]
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)

\[
y(k) = \mathbf{w}^T \varphi(k)
\]

Non-linear mapping
Fixed or trained supervised or unsupervised

Linear trainable layer

\[
\varphi(k) \mathbf{w}
\]
Radial basis function (RBF) network

- Network with one trainable layer

\[ g(x) = \sum_{j=1}^{M} \phi_j(x) \]

\[ \phi_j(x) = \exp \left( -\frac{(x - c_j)^2}{\sigma_j^2} \right) \]

Radial, e.g. Gaussian basis function
CMAC network

- Network with one trainable layer

\[ y = w^T a(x) \]
Feed-forward architecture

- Dynamic multi-layer network

\[ y^{(l-1)}_1 = x^{(l)}_1 \]

\[ y^{(l)}_1 = f(.) \]

\[ s_1^{(l)} = f(.) \]

\[ y^{(l)}_2 = x^{(l+1)}_2 \]

\[ s_2^{(l)} = f(.) \]

\[ y^{(l)}_2 = s_2^{(l)} \]

1-th layer

\[ y^{(l)}_1 = w^{(l)}_{21} \]

\[ s_1^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]

\[ y^{(l)}_2 = \sum \]

\[ s_2^{(l)} = \sum \]
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)

Input layer  1. hidden layer  2. hidden layer  Output layer

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

- General 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

\[ y(k) = f(x(k), x(k-1), \ldots, x(k-N)) \]
Basic neural network architectures

• Non-linear dynamic model structures based on regressor
  - NARX

\[ y(k) = f(x(k), \ldots, x(k-N), d(k-1), \ldots, d(k-M)) \]

From the system’s output, \( d(k) \)
Basic neural network architectures

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

\[ y(k) = f(x(k), \ldots, x(k-N), y(k-1), \ldots, y(k-M)) \]
Basic neural network architectures

• Non-linear dynamic model structures based on regressor

  - NARMAX

  \[ y(k) = f(x(k), \ldots, x(k-N), d(k-1), \ldots, d(k-M), \varepsilon(k-1), \ldots, \varepsilon(k-L)) \]

  - NJB

  \[ y(k) = f(x(k), \ldots, x(k-N), y(k-1), \ldots, y(k-M), \varepsilon(k-1), \ldots, \varepsilon(k-L), \varepsilon_x(k-1), \ldots, \varepsilon_x(k-K)) \]

  - NSS nonlinear state space representation
Basic neural network architectures

- Nonlinear function of regressor

\[ y(k) = f(w, \varphi(k)) \]

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

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

- nonlinear-in-the-parameter models

\[ y(k) = \sum_{j=1}^{n} w_j^{(2)} f_j(w^{(1)}, \varphi(k)) \quad w = [w_1^{(2)} w_2^{(2)} \ldots w_n^{(2)}, w^{(1)}]^T \]
Basic neural network architectures

• Basis functions $f_j(\varphi(k))$
  - MLP (with single nonlinear hidden layer)
    • sigmoidal basis function
      $$\text{sgm}(s) = \frac{1}{1 + e^{-Ks}}$$
      $$y(k) = \sum_{j=1}^{n} w_{j}^{(2)} f_j(w^{(1)}, \varphi(k))$$
      $$f_j(w^{(1)}, \varphi(k)) = \text{sgm}(\varphi(k)^T \mathbf{w}_j^{(1)} + \mathbf{w}_j^{(0)})$$
  - RBF (radial basis function, e.g. Gaussian)
    $$y(k) = \sum_j w_j f_j(\varphi(k)) = \sum_j w_j f(\|\varphi - \mathbf{c}_j\|)$$
    $$f(\varphi - \mathbf{c}_j) = \exp\left[-\|\varphi - \mathbf{c}_i\|^2 / 2\sigma_i^2\right]$$
  - 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 $f: \mathbb{R}^N \rightarrow \mathbb{R}$ on a compact subset $K$ of $\mathbb{R}^N$ 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.)

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

\[
\max_{x \in K} \left| f(x_1, \ldots, x_N) - \hat{f}(x_1, \ldots, x_N) \right| < \varepsilon \quad \varepsilon > 0
\]
Capability of networks

• Approximation of function (MLP)
  - Arbitrary continuous function $f: \mathbb{R}^N \rightarrow \mathbb{R}$ on a compact subset of $\mathbb{R}^N$ 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.)

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

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

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

\( K \) compact subset of \( R^N \)

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

\[ K = \bigcup_{j=1}^{k} X^{(j)} \text{ and } X^{(j)} \cap X^{(j)} \text{ is empty if } i \neq j \]
Capability of networks

• Universal approximator (RBF)

An arbitrary continuous function \( f : \mathbb{R}^N \rightarrow \mathbb{R} \) on a compact subset \( K \) of \( \mathbb{R}^N \) 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 \( g : \mathbb{R}^N \rightarrow \mathbb{R} \) 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

Space of possible input vectors

x→a

association vector

a

C=4

w

weight vector (trainable)

y

Σ

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

Number of basis functions

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

- C overlays
- C = 4
- Overlapping regions
- Quantization intervals
- Points of main diagonal
- Points of subdiagonal
- Regions of one overlay

Neural Networks for System Modeling • Gábor Horváth, 2005 Budapest University of Technology and Economics
CMAC approximation capability

Consistency equations:

$$f(a) - f(b) = f(c) - f(d)$$

can model only additive functions

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

  \[ C \text{ generalization parameter} \]
  \[ d_{\text{train}} \text{ distance between adjacent training data} \]

- Interesting behavior

  \[ C = l \times d_{\text{train}} : \text{linear interpolation between the training points} \]

  \[ C \neq l \times d_{\text{train}} : \text{significant generalization error} \]
  \[ \text{non-smooth output} \]
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
CMAC generalization error univariate case (max)

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

\[ y = \Phi x \quad \Phi = [\phi_1, \phi_2, ..., \phi_N]^T \quad \phi_i^T \phi_j = \delta_{ij}, \text{ further } \Phi^T \Phi = I, \quad \rightarrow \Phi^T = \Phi^{-1} \]

\[ x = \sum_{i=1}^{N} y_i \phi_i \quad \hat{x} = \sum_{i=1}^{M} y_i \phi_i, \quad M \leq N \]

\[ \varepsilon^2 = E\left\{ \|x - \hat{x}\|^2 \right\} = E\left\{ \left\| \sum_{i=1}^{N} y_i \phi_i - \sum_{i=1}^{M} y_i \phi_i \right\|^2 \right\} = \sum_{i=M+1}^{N} E\left\{ (y_i)^2 \right\} \]

\[ \hat{\varepsilon} = \varepsilon^2 - \sum_{i=M+1}^{N} \lambda_i \left( \phi_i^T \phi_i - 1 \right) = \sum_{i=M+1}^{N} \left[ \phi_i^T C_{xx} \phi_i - \lambda_i \left( \phi_i^T \phi_i - 1 \right) \right] \]

\[ C_{xx} = E\{xx^T\} \]

\[ \frac{\partial \hat{\varepsilon}}{\partial \phi_i} = \sum_{i=M+1}^{N} \left[ 2C_{xx} \phi_i - 2 \lambda_i \phi_i \right] = 0 \]

\[ C_{xx} \phi_i = \lambda_i \phi_i \quad \varepsilon^2 = \sum_{i=M+1}^{N} \phi_i^T C_{xx} \phi_i = \sum_{i=M+1}^{N} \phi_i^T \lambda_i \phi_i = \sum_{i=M+1}^{N} \lambda_i \]
Data compression, PCA networks

- Principal component analysis (Karhunen-Loève transformation)

\[ y = \Phi x \]
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-Leibler 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 Hyvärinam: Survey of Independent Component Analysis
References and further readings


Learning
Learning in neural networks

• Learning: parameter estimation

  - supervised learning, learning with a teacher
    \[ (x, y, d) \text{ training set: } \{x_i, d_i\}_{i=1}^P \]
  
  - unsupervised learning, learning without a teacher
    \[ x, y \]

  - analytical learning
Supervised learning

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

\[ d = f(x, n) \]

\[ y = f_M(x, w) \]

\[ C = C(\varepsilon) \]
Supervised learning

• Criterion function
  - quadratic criterion function:
    \[ C(d, y) = C(\varepsilon) = E \left\{ (d - y)^T (d - y) \right\} = E \left\{ \sum_j (d_j - y_j)^2 \right\} \]
  - other criterion functions
    • e.g. \( \varepsilon \) insensitive
  - regularized criterion functions: \( C(d, y) = C(\varepsilon) + \lambda C_R \)
    adding a penalty (regularization) term
Supervised learning

- Criterion minimization
  \[ \hat{w} = \arg \min_w C(d, y(w)) \]
- 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 \[ w(k + 1) = w(k) + \mu \varepsilon(k)x(k) \]
  - gradient methods \[ w(k + 1) = w(k) + \mu Q(-\nabla(k)) \]

- steepest descent \[ Q = I \]
- Newton \[ Q = R^{-1} \]
- Levenberg-Marquardt
\[ w(k + 1) = w(k) - H(w(k))^{-1}\nabla C(w(k)). \quad H \simeq E\{\nabla y(w)\nabla y(w)^T\} + \lambda \Omega \]
- conjugate gradient
\[ w(k + 1) = w(k) + \alpha_k g_k \quad g_j^T R g_k = 0 \quad \text{if} \quad j \neq k \]
Perceptron training

\[ x_0 = 1 \rightarrow w_0 \]
\[ x_1 \rightarrow w_1 \]
\[ x_2 \rightarrow w_2 \]
\[ x_N \rightarrow w_N \]

\[ s = w^T x \]

\[ y = \text{sgn}(s) \]

\[ w(k + 1) = w(k) + \mu \varepsilon(k)x(k) \]

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 \( \|x\| \leq M, \mu > 0 \).
Gradient method

• Analytical solution
  - linear-in-the parameter model
    \[ y(k) = w^T(k)x(k). \]
  - quadratic criterion function
    \[
    C(k) = E\left\{ \left( d(k) - w^T(k)x(k) \right)^2 \right\}
    \]
    \[
    = E\left\{ d^2(k) \right\} - 2E\left\{ d(k)x^T(k) \right\}w(k) + w^T(k)E\left\{ x(k)x^T(k) \right\}w(k)
    \]
    \[
    = E\left\{ d^2(k) \right\} - 2p^T w(k) + w^T(k)Rw(k)
    \]
  - Wiener-Hopf equation
    \[ w^* = R^{-1}p. \]
    \[ R = E\{ xx^T \} \]
    \[ p = E\{ xy \} \]
Gradient method

- Iterative solution

\[ w(k + 1) = w(k) + \mu(-\nabla(k)). \]

- gradient

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

- condition of convergence

\[ 0 < \mu < \frac{1}{\lambda_{\text{max}}} \quad \lambda_{\text{max}} : \text{maximal eigenvalue of } R \]
Gradient method

• LMS: iterative solution based on instantaneous error

\[ \varepsilon(k) = d(k) - x^T(k)w(k) \quad \hat{C}(k) = \varepsilon^2(k) \]

- instantaneous gradient

\[ \hat{\nabla}(k) = \frac{\partial \hat{C}(k)}{\partial w(k)} = 2\varepsilon(k) \frac{\partial \varepsilon(k)}{\partial w(k)} \]

- weight updating

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

- condition of convergence

\[ 0 < \mu < \frac{1}{\lambda_{max}} \]
Gradient methods

• Example of convergence

\[ w(0) \]

a.) small \( \mu \)  
b.) large \( \mu \)  
c.) conjugate gradient

steepest descent
Gradient methods

- Single neuron with nonlinear activation function

\[ \varepsilon(k) = d(k) - y(k) = d(k) - \text{sgm}(s(k)) = d(k) - \text{sgm}(w^T(k)x(k)) \]

\[ w(k + 1) = w(k) + 2\mu(k)\varepsilon(k)\text{sgm}'(s(k))x(k) = w(k) + 2\mu(k)\delta(k)x(k) \]
Gradient methods

- Multi-layer network: error backpropagation (BP)

\[
\begin{align*}
\mathbf{w}_i^{(l)}(k + 1) &= \mathbf{w}_i^{(l)}(k) + 2\mu \left( \sum_{r=1}^{N_{l+1}} \delta_r^{(l+1)}(k) \mathbf{w}_{ri}^{(l+1)}(k) \right) \sigma'(s_i^{(l)}(k)) \mathbf{x}^{(l)}(k) \\
&= \mathbf{w}_i^{(l)}(k) + 2\mu \delta_i^{(l)}(k) \mathbf{x}^{(l)}(k)
\end{align*}
\]

\[
\delta_i^{(l)}(k) = \left( \sum_{r=1}^{N_{l+1}} \delta_r^{(l+1)}(k) \mathbf{w}_{ri}^{(l+1)}(k) \right) \sigma'(s_i^{(l)}(k))
\]

\(l = \) 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, $\mu$
  - 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

Increasing the number of hidden units decreasing

Number of training cycles

C training error

(a)
(b)
(c)(d)(e)
(f)
Designing of an MLP

- Generalization, overfitting

Proper fitting to training points

Generalization

Input

Training points

Overfitting

Output
Designing of an MLP

- Early stopping for avoiding overfitting
Designing of an MLP

• Regularization
  - parametric penalty
    \[ C_r(w) = C(w) + \lambda \sum_{i,j} |w_{ij}| \]
    \[ \Delta w_{ij} = \mu \left( -\frac{\partial C}{\partial w_{ij}} \right) - \mu \lambda \text{sgn}(w_{ij}) \]
    \[ C_r(w) = C(w) + \lambda \sum_{|w_{ij}| \leq \Theta} |w_{ij}| \]
  - nonparametric penalty
    \[ C_r(w) = C(w) + \lambda \Phi(\hat{f}(x)) \]
    where \( \Phi(\hat{f}(x)) \) is some measure of smoothness
MLP as linear data compressor

- Subspace transformation

![Diagram of MLP as linear data compressor](image.png)
Nonlinear data compression

Nonlinear linear nonlinear linear
layer
RBF (Radial Basis Function)

\[ y = \sum_{i} w_i g_i(x) = \sum_{i} w_i g(\|x - c_i\|) = w^T x \quad g_i(x) = \exp\left[-\frac{\|x - c_i\|^2}{2\sigma_i^2}\right] \]

input layer  hidden layer  output layer

- \( g_1(x) = \varphi_1 \)
- \( g_2(x) = \varphi_2 \)
- \( g_M(x) = \varphi_M \)
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, $\mu$
  - 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
CMAC network

- Network with hash-coding
CMAC modeling capability

- Analytical solution
- Iterative algorithm (LMS)

\[ y(u_i) = a(u_i)^T w \quad i = 1, 2, \ldots, P \quad y = Aw \]

\[ w^* = A^\dagger d \quad A^\dagger = A^T (AA^T)^{-1} \]

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
  \[ \Delta w_{i*} = \mu(x - w_{i*}) \]

\[ w_{i*}^T x \geq w_i^T x \quad \forall i \]
PCA network

- Oja rule

It can be proofed: $w$ converges to the largest eigenvector.
PCA network

- Oja rule as a maximum problem (gradient search)

\[ y = w^T x \]

\[ f(w) = \frac{E\{y^2\}}{w^T w} = \frac{w^T R w}{w^T w} \]

\[ \nabla_{f(w)} = 2Rw - (w^T R w)2w \]

\[ \nabla_{f(w)} = 2E\{xx^T\}w - 2E\{w^T xx^T w\}w \]

\[ = 2E\{xy\} - 2E\{y^2\}w \]

\[ \nabla_{f(w)} = 2xy - 2y^2w = 2y(x - wy) \]

\[ w(k + 1) = w(k) + \mu y(k)[x(k) - w(k)y(k)] \]

Solution: gradient method with instantaneous gradient
PCA networks

- GHA network (Sanger network)

\[ \Delta w_i = \mu \left( y_i x^{(1)} - y_i^2 w_i \right) \]

\[ x^{(2)} = x^{(1)} - (w_1^T x^{(1)})w_1 = x^{(1)} - y_1 w_1 \]

\[ \Delta w_2 = \mu \left( y_2 x^{(2)} - y_2^2 w_2 \right) = \mu \left( y_2 x^{(1)} - y_1 y_2 w_1 - y_2^2 w_2 \right) \]

\[ \Delta w_i = \mu \left( y_i x^{(1)} - y_i^2 w_i \right) = \mu \left( y_i x^{(1)} - y_1 y_2 w_i - \ldots - y_i^2 w_i \right) = \mu \left( y_i x^{(1)} - \sum_{j=1}^{i-1} y_i y_j w_i - y_i^2 w_i \right) \]

\[ \Delta W = \eta \left[ yx^T - LT(yyyy^T)W \right] \]

Oja rule + Gram-Schmidt orthogonalization
PCA networks

- Oja rule for multi-output (subspace problem)

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

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-Leibler 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)

\[ x(k) = As(k) + n(k) = \sum_{i=1}^{M} a_i s_i(k) + n(k) \quad \quad \quad y(k) = Bx(k) = \hat{s}(k) \]

First step: whitening

\[ v(k) = Vx(k) \quad \quad \quad E\{v(k)v(k)^T\} = I \]

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

Second step: separation

\[ C(y) = \sum_{i=1}^{M} \left| \text{cum}(y_i^4) \right| = \sum_{i=1}^{M} \left| E\{y_i^4\} - 3E\{y_i^2\}^2 \right| \]

\[ B(k) = W(k)V(k) \]

\[ W(k + 1) = W(k) + \eta(k)[v(k) - W(k)g(y(k))] g(y^T(k)) \quad \quad \quad g(t) = \tanh(t) \]
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

\[ x(k) = As(k) + n(k) = \sum_{i=1}^{M} a_i s_i(k) + n(k) \]

\[ y(k) = Bx(k) = \hat{s}(k) \]

ICA network architecture
References and further readings


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 (sequential 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

\[ \varepsilon^2 = \sum_{k=1}^{K} \varepsilon^2(k) \]

\[ \frac{\partial \varepsilon^2}{\partial \mathbf{w}_{ij}^{(l)}} = \sum_k \frac{\partial \varepsilon^2}{\partial \mathbf{w}_{ij}^{(l)}} \]

\[ \frac{\partial \varepsilon^2}{\partial \mathbf{w}_{ij}^{(l)}} = \sum_k \frac{\partial \varepsilon^2}{\partial \mathbf{s}_i^{(l)}(k)} \frac{\partial \mathbf{s}_i^{(l)}(k)}{\partial \mathbf{w}_{ij}^{(l)}} \]

- output layer

\[ \mathbf{w}_{ij}^{(L)}(k+1) = \mathbf{w}_{ij}^{(L)}(k) + 2\mu \varepsilon_i f'(s_i^{(L)}(k)) x_j^{(L)}(k) \]

- hidden layer

\[ \mathbf{w}_{ij}(k+1) = \mathbf{w}_{ij}(k) + 2\mu \delta_i (k-M) x_j(k-M) \]

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

\[ \Delta_m^{(l)}(k) = \left[ \delta_m^{(l)}(k), \delta_m^{(l)}(k+1), \ldots, \delta_m^{(l)}(k + M_{m}(l)) \right] \]
Recurrent network

- Architecture

![Diagram of Recurrent Network with inputs $x_1(k), x_2(k), \ldots, x_N(k)$ and outputs $y_1(k+1), y_2(k+1), \ldots, y_M(k+1)$ connected through a feed-forward (static) network with weights $W$. The diagram includes delay elements $z^{-1}$ for feedback connections.]
Recurrent network

- Training: real-time recursive learning (RTRL)

\[
\frac{\partial \varepsilon^2(k)}{\partial w_{ij}(k)} = \sum_{l \in \mathcal{C}} \frac{\partial \varepsilon^2_l(k)}{\partial w_{ij}(k)}; \quad \Delta w_{ij}(k) = -\mu \frac{\partial \varepsilon^2(k)}{\partial w_{ij}(k)}; \quad \frac{\partial \varepsilon^2(k)}{\partial w_{ij}(k)} = 2 \sum_{l \in \mathcal{C}} \varepsilon_l(k) \frac{\partial \varepsilon_l(k)}{\partial w_{ij}(k)} = -2 \sum_{l \in \mathcal{C}} \varepsilon_l(k) \frac{\partial y_l(k)}{\partial w_{ij}(k)}
\]

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

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

\[
u_i(k) = \begin{cases} x_i(k) & \text{if } i \in \mathcal{A} \\ y_i(k) & \text{if } i \in \mathcal{B} \end{cases}
\]

\[
\varepsilon_i^2(k) = \begin{cases} (d_i(k) - y_i(k))^2 & \text{if } i \in \mathcal{C}(k) \\ 0 & \text{otherwise} \end{cases}
\]
Recurrent network

- Training: backpropagation through time (BPTT) unfolding in time
Dynamic neural structures

- Combined linear dynamic and non-linear dynamic architectures

\[
\begin{align*}
\frac{\partial \varepsilon(k)}{\partial w_{ij}} &= \frac{\partial y(k)}{\partial w_{ij}} = H(z) \frac{\partial v}{\partial w_{ij}} & \frac{\partial \varepsilon(k)}{\partial w_{ij}^{(2)}} &= \frac{\partial y(k)}{\partial w_{ij}^{(2)}} = \sum_l \frac{\partial y(k)}{\partial v_l} \frac{\partial v_l}{\partial w_{ij}^{(2)}}
\end{align*}
\]

Dynamic backpropagation
Dynamic neural structures

a.)

\[
\frac{\partial \varepsilon(k)}{\partial w_{ij}} = \frac{\partial y(k)}{\partial w_{ij}} = \frac{\partial N(v)}{\partial v} H(z) \frac{\partial y(k)}{\partial w_{ij}} + \frac{\partial N(v)}{\partial w_{ij}}
\]

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

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

- 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


Support vector machines
Outline

• Why we need a new approach
• 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

Optimal hyperplane

Margin
Support vector machines

- Linearly separable two-class problem

\[
\{ (x_i, y_i) \}_{i=1}^P \quad x_i \in X^1 \quad y_i = +1, \quad x_i \in X^2 \quad y_i = -1
\]

separating hyperplane

\[
w^T x + b = 0
\]

\[
w^T x_i + b \geq +1, \text{ if } x_i \in X^1
\]

and

\[
w^T x_i + b \leq -1, \text{ if } x_i \in X^2
\]

\[
(w^T x_i + b)y_i \geq 1, \quad \forall i
\]
Support vector machines

• Geometric interpretation

The formulation of optimality

\[
d(w, b, x) = \frac{|w^T x + b|}{\|w\|}
\]

\[
\rho(w, b) = \min_{\{x_i; y_i=1\}} d(w, b, x_i) + \min_{\{x_i; y_i=-1\}} d(w, b, x_i) =
\]

\[
= \min_{\{x_i; y_i=1\}} \frac{|w^T x_i + b|}{\|w\|} + \min_{\{x_i; y_i=-1\}} \frac{|w^T x_i + b|}{\|w\|} = \frac{2}{\|w\|}
\]
Support vector machines

• Criterion function (primal problem)

\[ \min \|w\|^2 \rightarrow \max \text{ margin} \]

\[ \Phi(w) = \frac{1}{2} \|w\|^2 \quad \text{with the conditions} \quad (w^T x_i + b) y_i \geq 1, \quad \forall i \]

a constrained optimization problem

(KKT conditions, saddle point)

\[ J(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{P} \alpha_i \{[w^T x_i + b] y_i - 1\} \quad \max \min_{\alpha, w, b} J(w, b, \alpha) \]

conditions

\[ \frac{\partial J}{\partial w} = w - \sum_{i=1}^{P} \alpha_i x_i y_i = 0 \quad \frac{\partial J}{\partial b} = \sum_{i=1}^{P} \alpha_i y_i = 0 \quad w = \sum_{i=1}^{P} \alpha_i x_i y_i \quad \sum_{i=1}^{P} \alpha_i y_i = 0 \]
Support vector machines

- Lagrange function (dual problem)

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

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

support vectors

\[
\mathbf{x}_i : \alpha_i > 0
\]

optimal hyperplane

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

output

\[
y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \sum_{i=1}^{P} \alpha_i y_i (\mathbf{x}_i^T \mathbf{x}) + b
\]
Support vector machines

- Linearly nonseparable case (slightly nonlinear case)

Separating hyperplane

\[ y_i[w^T x_i + b] \geq 1 - \xi_i \quad i = 1, \ldots, P \]

criterion function (slack variable \( \xi \))

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

Lagrange function

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

\[ 0 \leq \alpha_i \leq C \]

Support vectors \( x_i : \alpha_i > 0 \)  

Optimal hyperplane \( w^* = \sum_{i=1}^{P} \alpha_i y_i x_i \)
Support vector machines

• Nonlinear separation, feature space
  - separating hypersurface (hyperplane in the $\phi$ space)
    $\mathbf{w}^T \phi(\mathbf{x}) + b = 0$
    $\sum_{j=0}^{M} w_j \phi_j(\mathbf{x}) = 0$
  - decision surface
    $\sum_{i=1}^{P} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) = \sum_{i=1}^{P} \left( \alpha_i y_i \sum_{j=0}^{M} \phi_j(\mathbf{x}_i) \phi_j(\mathbf{x}) \right) = 0$
  - kernel function (Mercer conditions)
    $K(\mathbf{x}_i, \mathbf{x}_j) = \phi^T(\mathbf{x}_i) \phi(\mathbf{x}_j)$
  - 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

Separating curve

(higher dimensional) Feature (ϕ) space

Separating plane

\[ y = \sum_{j=0}^{M} w_j \varphi_j(x) = w^T \varphi(x) \]

\[
\begin{align*}
\mathbf{w} &= [w_0, w_1, \ldots, w_M]^T \\
\varphi &= [\varphi_0(x), \varphi_1(x), \ldots, \varphi_M(x)]^T
\end{align*}
\]
Kernel space

- Kernel trick

feature representation (nonlinear transformation) is not used
kernel function values (scalar values are used)

\[
K(x_i, x_j) = \phi^T(x_i)\phi(x_j)
\]

\[
y(x) = \mathbf{w}^* \phi(x) + b = \sum_{i=1}^{p} \alpha_i y_i \left( \phi^T(x_i)\phi(x) \right) + b = \sum_{i=1}^{p} \alpha_i y_i K(x_i, x) + b
\]
Support vector machines

- Examples of kernel functions

  - Polynomial
    
    \[ K(x, x_i) = (x^T x_i + 1)^d, \quad d = 1, \ldots \]

  - RBF
    
    \[ K(x, x_i) = \exp\left(-\frac{1}{2\sigma^2} \|x - x_i\|^2\right) \]

  - MLP (only for certain \( \beta_0 \) and \( \beta_1 \))
    
    \[ K(x, x_i) = \tanh(\beta_0 x^T x_i + \beta_1) \]

  - CMAC B-spline
Support vector machines

• Example: polynomial basis and kernel function
  - basis functions

  \[ \varphi(x_i) = [1, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}]^T \]

  - kernel function

  \[ K(x, x_i) = 1 + x_1^2 x_{i1}^2 + 2x_1 x_2 x_{i1} x_{i2} + x_2^2 x_{i2}^2 + 2x_1 x_{i1} + 2x_2 x_{i2} \]
SVR (regression)

\[ C_\varepsilon (y, f(x, \alpha)) = \begin{cases} 0 & \text{ha } |y - f(x, \alpha)| \leq \varepsilon \\ |y - f(x, \alpha)| - \varepsilon & \text{otherwise} \end{cases} \]
SVR (regression)

\[ f(x) = \sum_{j=0}^{M} w_j \varphi_j(x) \]

Constraints:

\[ y_i - w^T \varphi(x_i) \leq \varepsilon + \xi_i , \]

\[ w^T \varphi(x_i) - y_i \leq \varepsilon + \xi'_i , \]

\[ \xi_i \geq 0 , \]

\[ \xi'_i \geq 0 , \]

Minimize:

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

Lagrange function

\[
J(w, \xi, \xi', \alpha, \alpha', \gamma, \gamma') = C \sum_{i=1}^{P} (\xi_i + \xi_i') + \frac{1}{2} w^T w - \sum_{i=1}^{P} \alpha_i [w^T \phi(x_i) - y_i + \varepsilon + \xi_i] -
\]

\[
- \sum_{i=1}^{P} \alpha'_i [y_i - w^T \phi(x_i) + \varepsilon + \xi'_i] - \sum_{i=1}^{P} (\gamma_i \xi_i + \gamma'_i \xi'_i)
\]

\[
\gamma_i = C - \alpha_i \quad \gamma'_i = C - \alpha'_i
\]
SVR (regression)

- Dual problem

$$W(\alpha, \alpha') = \sum_{i=1}^{P} y_i (\alpha_i - \alpha'_i) - \epsilon \sum_{i=1}^{P} (\alpha_i + \alpha'_i) - \frac{1}{2} \sum_{i=1}^{P} \sum_{j=1}^{P} (\alpha_i - \alpha'_i)(\alpha_j - \alpha'_j)K(x_i, x_j)$$

constraints

$$\sum_{i=1}^{P} (\alpha_i - \alpha'_i) = 0 \quad 0 \leq \alpha_i \leq C, \quad 0 \leq \alpha'_i \leq C, \quad x_i : \alpha_i \neq \alpha'_i$$

solution

$$w^* = \sum_{i=1}^{P} (\alpha_i - \alpha'_i) \phi(x_i)$$

$$y(x) = w^T \phi(x) = \sum_{i=1}^{P} (\alpha_i - \alpha'_i)(\phi^T(x_i)\phi(x)) = \sum_{i=1}^{P} (\alpha_i - \alpha'_i)K(x, x)$$
SVR (regression example)
SVR (regression example)
SVR (regression)

$\varepsilon = 0.1$
Support vector machines

- Main advantages
  - automatic model complexity (network size)
  - relevant training data point selection
  - allows tolerance ($\varepsilon$)
  - 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 ($C$, $\varepsilon$, $\sigma$) selection
  - batch processing (there are on-line versions too)
SVM versions

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

classification

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

regression

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

equality constraints

\[ y_i [w^T x_i + b] = 1 - e_i \quad i = 1, \ldots, P \]

\[ y_i = w^T \phi(x_i) + b + e_i \quad i = 1, \ldots, P \]

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

- Ridge regression

  similar to LS-SVM
LS-SVM

Lagrange equation

\[
L(w, b, e; \alpha) = \frac{1}{2} w^T w + \frac{1}{2} \gamma \sum_{k=1}^{P} e_k^2 - \sum_{k=1}^{P} \alpha_k \left( w^T \varphi(x_k) + b + e_k - y_k \right)
\]

The results

\[
\frac{\partial L}{\partial w} = 0 \quad \rightarrow \quad w = \sum_{k=1}^{P} \alpha_k \varphi(x_k)
\]

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

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

\[
\frac{\partial L}{\partial \alpha_k} = 0 \quad \rightarrow \quad w^T \varphi(x_k) + b + e_k - y_k = 0 \quad \quad \quad \quad k = 1, \ldots, P
\]
**LS-SVM**

Linear equation

Regression

\[
\begin{bmatrix}
0 & \Omega_{+1}^T \\
\Omega & \Omega_{+1}
\end{bmatrix}
\begin{bmatrix}
b \\
a
\end{bmatrix}
= \begin{bmatrix}
y \\
0
\end{bmatrix}
\]

where

\[
\Omega_{i,j} = K(x_i, x_j) = \varphi^T(x_i)\varphi(x_j)
\]

Clasification

\[
\begin{bmatrix}
0 & y^T \\
y & \Omega_{+y^{-1}}
\end{bmatrix}
\begin{bmatrix}
b \\
a
\end{bmatrix}
= \begin{bmatrix}
y_1 \\
0
\end{bmatrix}
\]

the response of the network

\[
y(x) = \sum_{k=1}^{N} \alpha_k K(x, x_k) + b
\]

\[
y(x) = \sum_{k=1}^{N} \alpha_k y_k K(x, x_k) + b
\]
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 sparse solution, all training points are used (there are no “support vectors”)
  - No “tolerance parameter” ($\varepsilon$)
  - 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)
  - Details: see the references
- Additional constraints 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)

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

\[
w_i(k+1) = w_i(k) + \mu(k)e(k) + \lambda \left( \frac{y_d(k)}{C} - w_i(k) \right)
\]
Kernel CMAC

- Classical Albus CMAC: analytical solution

\[ y_{d_k} = w^T a_k \quad k = 1, \ldots, P \quad y_d = Aw \]

\[ w^* = A^\dagger y_d \quad A^\dagger = A^T (AA^T)^{-1} \quad y(x) = a^T (x) w^* = a^T (x) A^T (AA^T)^{-1} y_d \]

- Kernel version

Criterion function (LS)

\[ \min_w J(w, e) = \frac{1}{2} w^T w + \gamma \sum_{k=1}^{P} e_k^2 \]

Constraint

\[ y_{d_k} = w^T a_k + e_k \]

Lagrangian

\[ L(w, e, \alpha) = J(w, e) - \sum_{k=1}^{P} \alpha_k \left( w^T a_k + e_k - y_{d_k} \right) \]
Kernel CMAC (ridge regression)

- Using the derivatives the resulted equations

\[
\begin{align*}
\frac{\partial L(w, e, \alpha)}{\partial w} &= 0 \quad \Rightarrow \quad w = \sum_{k=1}^{P} \alpha_k a_k \\
\frac{\partial L(w, e, \alpha)}{\partial e_k} &= 0 \quad \Rightarrow \quad \alpha_k = \gamma e_k \quad \quad k = 1, \ldots, P \\
\frac{\partial L(w, e, \alpha)}{\partial \alpha_k} &= 0 \quad \Rightarrow \quad w^T a(x_k) + e_k - y_{d_k} = 0 \quad k = 1, \ldots, P
\end{align*}
\]

\[
\begin{bmatrix} K + \frac{1}{\gamma} I \end{bmatrix} \alpha = y_d \quad \quad K = AA^T
\]

\[
y(x) = a^T(x)w = a^T(x)\sum_{k=1}^{P} \alpha_k a_k = \sum_{i=1}^{P} \alpha_k K(x, x_k) = K^T(x)\alpha
\]
Kernel CMAC with regularization

Extended criterion function:

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

Lagrange function

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

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

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

Output

$$y(x) = a^T (x) (I + \lambda D)^{-1} \left[ a + \frac{\lambda}{C} y_d \right] \quad \text{where} \quad D = \sum_{k=1}^{P} \text{diag}(a_k)$$
Kernel CMAC with regularization

- Kernel function for two-dimensional kernel CMAC
Regularized Kernel CMAC (example)

- 2D sinc

Kernel network output over test mesh $c=4$, $d=3$, $m=32$, $mse=0.0028965$

without regularization

with regularization
References and further readings


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

\[ R(w) = \int l(x, w) p(x, y) dx dy = \int [y - f(x, w)]^2 p(x, y) dx dy \quad R(w^* | P) \]

• Difficulties: joint density function is unknown

Only the empirical risk can be determined

\[ R_{\text{emp}}(w) = \frac{1}{P} \sum_{l=1}^{P} \left[ y_i - f(x_l, w) \right]^2 \quad R_{\text{emp}}(w^* | P) \]

optimal value

minimizing the empirical risk
Statistical learning theory: ERM principle

- Asymptotic consistency of empirical risk

\[ R_{\text{emp}}(w^*|P) \rightarrow R(w^\circ) \quad \text{when} \quad P \rightarrow \infty \]

\[ R(w^*|P) \rightarrow R(w^\circ) \quad \text{when} \quad P \rightarrow \infty \]
Statistical learning theory

• Condition of consistency of the ERM principle
  Necessary 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

![Diagram showing linear separation and 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 $1-\eta$ (confidence level; $\eta$ is a given value within the additional term)

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

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

$R =$ Radius of a sphere containing all data points

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

- regression

$$R(w) \leq \frac{R_{\text{emp}}(w)}{\left(1-c\sqrt{\varepsilon(h)}\right)_+} \quad \varepsilon = a_1 \frac{h[\log(a_2N / h)+1] - \log(\eta / 4)}{N}$$
Generalization error

\[ f(VC, \ldots) \]

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

\[ P / h \quad \text{small} \quad \Rightarrow \quad R \gg R_{\text{emp}} \]

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 $h_k$
  $S_1 \subset S_2 \subset ... \subset S_k \subset ...$

  The ordering of complexity of the elements
  $h_1 \leq h_2 \leq ... \leq h_k \leq ...$

  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 guaranteed 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 $S_k$

\[
R(w^k_P) \leq R_{\text{emp}}(w^k_P) + \text{additional term}(P/h_k) \leq R_{\text{emp}}(w^k_P) + \text{confidence interval}
\]
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


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

\[ \bar{y}(x, \alpha) = \sum_{j=0}^{M} \alpha_j y_j(x) \]
Ensemble of networks

• Mathematical justification

  – Ensemble output
    \[
    \bar{y}(x, \alpha) = \sum_{j=0}^{M} \alpha_j y_j(x)
    \]

  – Ambiguity (diversity)
    \[
    a_j(x) = \left[ y_j(x) - \bar{y}(x, \alpha) \right]^2
    \]

  – Individual error
    \[
    \epsilon_j(x) = \left[ d(x) - y_j(x) \right]^2
    \]

  – Ensemble error
    \[
    \epsilon(x) = \left[ d(x) - \bar{y}(x, \alpha) \right]^2
    \]

  – Constraint
    \[
    \sum_{j=1}^{M} \alpha_j = 1
    \]
Ensemble of networks

- Mathematical justification (cont’d)
  - Weighted error
    \[ \bar{\varepsilon}(x, \alpha) = \sum_{j=0}^{M} \alpha_j \varepsilon_j(x) \]
  - Weighted diversity
    \[ \bar{a}(x, \alpha) = \sum_{j=0}^{M} \alpha_j a_j(x) \]
  - Ensemble error
    \[ \varepsilon(x) = [d(x) - \bar{y}(x, \alpha)]^2 = \bar{\varepsilon}(x, \alpha) - \bar{a}(x, \alpha) \]
  - Averaging over the input distribution
    \[
    E = \int_{x} \varepsilon(x, \alpha) f(x) dx \quad \bar{E} = \int_{x} \bar{\varepsilon}(x, \alpha) f(x) dx \quad A = \int_{x} a(x, \alpha) f(x) dx
    \]
    \[ E = \bar{E} - \bar{A} \]

*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
  - \( \alpha_k = 1, \quad \alpha_j = 0, \quad j \neq k \), \( k \) depends on the input
    for different input domains different network (alone)
    gives the output
- optimal values using the constraint
  \( \sum_{k=1}^{M} \alpha_k = 1 \)
- optimal values without any constraint
  Wiener-Hopf equation
  \[ \alpha^{*}_{(1)} = R_{y}^{-1}P \]
  \[ R_{y} = E[\bar{y}(x) \bar{y}(x)^T] \]
  \[ P = E[\bar{y}(x) d(x)] \]
References and further readings


Gasser Auda and Mohamed Kamel: „Modular Neural Networks: A survey” Pattern Analysis and Machine 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

\[ \mu = \sum_{i=1}^{M} g_i \mu_i \quad \mu_i = f(x, \Theta_i) \quad \sum_{i=1}^{M} g_i = 1 \quad g_i \geq 0 \quad \forall i \]

\( \Theta_i \) is the parameter of the \( i \)-th expert

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

\[ g_i = \frac{e^{\xi_i}}{\sum_{j=1}^{M} e^{\xi_j}} \quad \xi_i = v_i^T x \]

• \( v_i^T \) is the parameter of the gating network
Mixture of Experts (MOE)

- Probabilistic interpretation

\[ \mu_i = E[y \mid x, \Theta_i] \quad g_i = P(i \mid x, v_i) \]

the probabilistic model with true parameters

\[ P(y \mid x, \Theta^0) = \sum_i g_i(x, v_i^0)P(y \mid x, \Theta_i^0) \]

a priori probability \[ g_i(x, v_i^0) = P(i \mid x, v_i^0) \]
Mixture of Experts (MOE)

- Training

  - Training data

    \[ X = \left\{ \left( x^{(l)}, y^{(l)} \right) \right\}_{l=1}^P \]

  - Probability of generating output from the input

    \[
    P(y^{(l)} | x^{(l)}, \Theta) = \sum_i P(i | x^{(l)}, v_i) P(y^{(l)} | x^{(l)}, \Theta_i)
    \]

    \[
    P(y | x, \Theta) = \prod_{l=1}^P P(y^{(l)} | x^{(l)}, \Theta) = \prod_{l=1}^P \left[ \sum_i P(i | x^{(l)}, v_i) P(y^{(l)} | x^{(l)}, \Theta_i) \right]
    \]

  - The log likelihood function (maximum likelihood estimation)

    \[
    L(x, \Theta) = \sum_l \log \left[ \sum_i P(i | x^{(l)}, v_i) P(y^{(l)} | x^{(l)}, \Theta_i) \right]
    \]
Mixture of Experts (MOE)

- **Training (cont’d)**
  - Gradient method
    \[
    \frac{\partial L(x, \Theta)}{\partial \Theta_i} = 0 \quad \text{and} \quad \frac{\partial L(x, \Theta)}{\partial v_i} = 0
    \]
  - The parameter of the expert network
    \[
    \Theta_i(k + 1) = \Theta_i(k) + \eta \sum_{l=1}^{P} h_i^{(l)} (y^{(l)} - \mu) \frac{\partial \mu_i}{\partial \Theta_i}
    \]
  - The parameter of the gating network
    \[
    v_i(k + 1) = v_i(k) + \eta \sum_{l=1}^{P} \left( h_i^{(l)} - g_i^{(l)} \right) x^{(l)}
    \]
Mixture of Experts (MOE)

- Training (cont’d)
  - A priori probability
    \[ g_i^{(l)} = g_i(x^{(l)}, v_i) = P(i \mid x^{(l)}, v_i) \]
  - A posteriori probability
    \[ h_i^{(l)} = \frac{g_i^{(l)} P(y^{(l)} \mid x^{(l)}, \Theta_i)}{\sum_j g_j^{(l)} P(y^{(l)} \mid x^{(l)}, \Theta_j)} \]
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 (Classification 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
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 (Classification And Regression Tree) for initial hierarchical MOE (HMOE) architecture and for the initial expert and gating network parameters
References and further readings


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 (Dunafer 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 features 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 - 10 ... +15 °C)
  - 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

System

Model

Inverse Model

Copy of Model

temperature

predicted temperature

model output temperature

predicted oxygen

oxygen

components (parameters)

components (parameters)

measured temperature

ε

ε
Overview

- Introduction
- Modeling approaches
- Building neural models
- Data base construction
- Model selection
- Modular approach
- Hybrid approach
- Information system
- Experiences with the advisory system
- Conclusions
The modeling task
„Neural” solution

- The steps of solving a practical problem

1. **Preprocessing**
2. **Neural network**
3. **Postprocessing**

Raw input data → Preprocessing → Neural network → Postprocessing → Results
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

Initial database

Neural network training

Sensitivity analysis

Input parameter of small effect on the output?

New database

Input parameter cancellation

yes

no
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

\[ \bar{x}_i = \frac{1}{P} \sum_{p=1}^{P} x_i^{(p)} \quad \sigma_i^2 = \frac{1}{P-1} \sum_{p=1}^{P} (x_i^{(p)} - \bar{x}_i)^2 \quad \tilde{x}_i^{(p)} = \frac{x_i^{(p)} - \bar{x}_i}{\sigma_i} \]

- Normalization into [0,1]

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

- Decorrelation + normalization

\[ \Sigma = \frac{1}{P-1} \sum_{p=1}^{P} (x^{(p)} - \bar{x})(x^{(p)} - \bar{x})^T \quad \Sigma \phi_j = \lambda_j \phi_j \quad \Lambda = \text{diag}(\lambda_1...\lambda_N) \]

\[ \tilde{x}^{(p)} = \Lambda^{-1/2} \Phi^T (x^{(p)} - \bar{x}) \quad \Phi = \begin{bmatrix} \phi_1 & \phi_2 & \cdots & \phi_N \end{bmatrix}^T \]
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 $x_j$ 
  \[ \tilde{C}(i, j) = \frac{C(i, j)}{\sqrt{C(i, i) C(j, j)}} \]
  previous (other) values 
  \[ \hat{x}_i = \bar{x}_i + \sigma_i \xi \]
  other parameters 
  \[ \hat{x}_i^{(k)} = \hat{f}(x_j^{(k)}) \quad \text{or} \quad \hat{x}_i^{(k)} = \hat{f}(x_j^{(k)}, x_j^{(k)}, \ldots) \]
  time dependence (dynamic problem) 
  \[ R_i(t, \tau) = E\{x_i(t) x_i(t + \tau)\} \]

- 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
  - $\varepsilon$-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 $\varepsilon$-insensitive loss function
- Regularization: example regularized (kernel) CMAC
Reducing the effect of input and output noise

- Errors in variables (EIV)

\[
\bar{x}_k = \frac{1}{M} \sum_{i=1}^{M} x_{k}^{[i]}
\]

\[
\sigma^2_{x,k} = \frac{1}{M-1} \sum_{i=1}^{M} (x_{k}^{[i]} - \bar{x}_k)^2
\]

\[
\bar{y}_k = \frac{1}{M} \sum_{i=1}^{M} y_{k}^{[i]}
\]

\[
\sigma^2_{y,k} = \frac{1}{M-1} \sum_{i=1}^{M} (y_{k}^{[i]} - \bar{y}_k)^2
\]
EIV

• LS vs EIV criterion function

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

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

• EIV training

\[ \Delta W_j = \eta \frac{M}{2P} \sum_{k=1}^{P} e_{f,k} \frac{\partial f_{NN}(x_k, W)}{\partial W_j} \]

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

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

• 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) → 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

\[ y_M(k) = f[x(k), x(k-1), x(k-2), ..., x(k-n), y(k-1), y(k-2), ..., y(k-m)] \]

Model order: the input dimension of the static network
Model order selection

- AIC, MDL, NIC, Lipschitz number

\[ y(k) = f[x(k), x(k-1), x(k-2), \ldots, x(k-n), y(k-1), y(k-2), \ldots, y(k-m)] \]

- Lipschitz number, Lipschitz quotient

\[ q^{(l)} = \left( \prod_{k=1}^{p} \sqrt{n} q^{(l)}(k) \right)^{1/p} \]

\[ q_{ij} = \frac{|y_i - y_j|}{|x_i - x_j|}, \]

- Noiseless case
- Noisy case
Model order selection

- Lipschitz quotient

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

  \[
  y = f [x_1, x_2, \ldots, x_n]
  \]

  bounded derivatives

  \[
  |f'_i| = \left| \frac{\partial f}{\partial x_i} \right| \leq M \quad i = 1, 2, \ldots, n
  \]

  Lipschitz quotient

  \[
  q_{ij} = \left| \frac{y_i - y_j}{x_i - x_j} \right|, \quad i \neq j \quad 0 \leq q_{ij} \leq L
  \]

  Sensitivity analysis

  \[
  \Delta y = \frac{\partial f}{\partial x_1} \Delta x_1 + \frac{\partial f}{\partial x_2} \Delta x_2 + \cdots + \frac{\partial f}{\partial x_n} \Delta x_n = f'_1 \Delta x_1 + f'_2 \Delta x_2 + \cdots + f'_n \Delta x_n
  \]
Model order selection

- Noisy case
  Combined method of Lipschitz + EIV

Lipschitz quotient for noisy data

Diagram: Model creation process with Lipschitz and EIV algorithms. Initial estimation leads to Untrained Neural Network, which is trained to create a Trained Network. This network is then used for New Estimation. The process involves comparing results with Untrained Neural Network to refine the model order selection.
Correlation based model order selection

- Model order 2...4 because of practical problems
- Too many input components
- \((2...4) \times (\text{number of input components + outputs})\)
- Too large network
- Too few training data
- The problem of missing data
- Network size: cross-validation
References and further readings


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 model for the different working conditions
- Processing of special cases
- Depending on the distribution of input parameters
- 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

Input data preparatory expert system

Mixture of experts system

Output expert system

Correction term expert system

Oxygen prediction

No prediction (explanation)

Input data
The hybrid-neural system

Data preprocessing and correction

Neural Model

Data preprocessing

Input data
The hybrid-neural system

Conditional network running

Input data

Expert for selecting a neural model

NN

NN

NN

NN

NN

O₁

O₂

Oₖ
The hybrid-neural system

Output expert

Parallel network running - postprocessing

Input data
The hybrid-neural system

Iterative network running

Neural network running, prediction making

Result satisfactory

Modification of input parameters
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 conclusions

• 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


References and further readings


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


