s c c h
software competence center
hagenberg

FUZZY.LOGIC.LAB.LINZ

Abstracts of the FLLL/SCCH Master and PhD Seminar

Room 0/10, Software Park Hagenberg

November 11, 2005

Software Competence Center Hagenberg
Hauptstrasse 99
A-4232 Hagenberg
Tel. +43 7236 3343 800
Fax +43 7236 3343 888
www.scch.at

Fuzzy Logic Laboratorium Linz-Hagenberg
Hauptstrasse 99
A-4232 Hagenberg
Tel. +43 7236 3343 431
Fax +43 7236 3343 434
www.flll.jku.at

Program

Session 1 (Chair: Ulrich Bodenhofer) 8:30–10:10

8:30 Leila Muresan:
Denoising Fluoresence Microscopy Image Sequences

8:55 Rainer Geschray:
Elicitation of Control Strategies from Data-Based Fuzzy Models

9:20 Bernhard Moser:
Segmentation and Tracking by Mean Shift: A Tutorial

9:45 Radko Mesiar:
Copulas: New Results and Open Problems

10:10 Coffee Break

Session 2 (Chair: Leila Muresan) 10:30–12:10

10:30 Edwin Lughofer:
A Modified Version of Vector Quantization for Incremental Clustering

10:55 Endre Pap:
Actual Results and Open Problems in the Theory of Generalized Analysis

11:20 Bernhard Moser:
Characterization of Kernels in the Frequency Domain: A Tutorial

11:45 Werner Groißböck:
A Nonlinear Approximation Formula Generator for High Dimensional Data Based on Variable Selection and Genetic Programming

Technical Report
FLLL–TR–0210

FUZZY.LOGIC.LAB.LINZ

Denoising fluorescence microscopy image sequences

Leila Muresan
Fuzzy Logic Laboratorium Linz-Hagenberg
e-mail leila.muresan@jku.at

Johannes Kepler Universität Linz

Institut für Wissensbasierte Mathematische Systeme
Johannes Kepler Universität Linz
Altenbergerstrasse 69
A-4040 Linz
Austria

Fuzzy Logic Laboratorium Linz-Hagenberg

FLLL
Softwarepark Hagenberg
Hauptstrasse 99
A-4232 Hagenberg
Austria

Introduction

Denoising microscopy image sequences proves to be a difficult task given a set of problems arising from the imaging technique as well as from the typical content of the biological image sequences. The algorithms designed to process natural images make use of a series of interest points and features, like edges, corners, etc., which are not available in the case of intracellular images. The typical content of the latter are usually isotropic bright blobs, without a clear boundary.

In the case of image sequences, the task becomes even harder due to the motion of the imaged objects. In the case of natural images objects can be identified through images by using cues such as shape, color, texture, and denoising can be performed allowing for motion compensation. However these cues usually are not available in the case of biological "objects".

Another class of difficulties is generated by the imaging technique. Due to the quantum nature of light the image formation can be modeled as a Poisson process:

$$P(f(x)|u(x)) = \frac{e^{-u(x)}u(x)^{f(x)}}{f(x)!} \tag{1}$$

where $u(x) \in \mathbb{R}$ represent the unknown intensity values of the original image at each pixel x, while $f(x)$ are the (integer, 12 bit) measured intensities at x. Moreover, in case of *in-vivo* single protein imaging the number of photon count is low (due to short laser illumination in order to avoid bleaching).

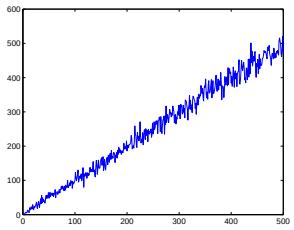

Figure 1: The function $f(x) = x$ corrupted by Poisson noise.

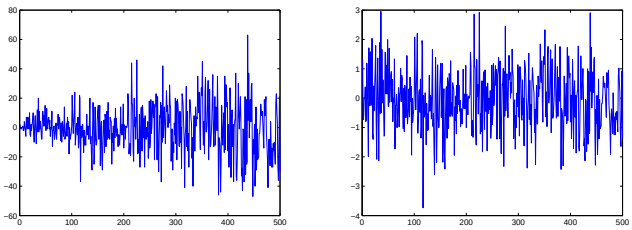

Figure 2: Error without and with Anscombe transform

One way to cope with these specific challenges consists of transforming the input image (or image sequence) such that well-known algorithms can be successfully applied. For instance, in order to cope with Poisson noise one can apply the Anscombe transform, as in [2]:

$$A\{f(x)\} = 2\sqrt{f(x) + \frac{3}{8}} \tag{2}$$

which modifies the data as if generated by a Gaussian noise model with $\sigma = 1$.

The denoising algorithms designed for Gaussian noise are applied on the transformed image and the inverse transform is performed on the result. Figures 1 and 2 explain the effects of the Poisson noise and of the Anscombe transform.

In the case of object motion, one can simply ignore the 3D nature of the problem and perform the denoising in 2D. However, as can be seen in fig. 3, the results are not always convincing.

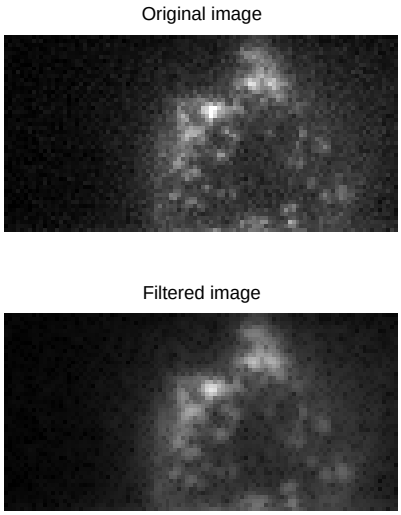

Figure 3: 2D denoising of a frame in the image sequence

In this paper we give a survey of popular denoising algorithms, adapted for 3D signals:

- Adaptive mean filter
- Anisotropic difussion
- Variational approach
- Spatial Tonal Convolution

1 Adaptive mean filter in 3D

The adaptive mean filter, [3], is defined as :

$$u(x) = \bar{u(x)} + \frac{V(x)}{V(x) + V_G}(u(x) - \bar{u(x)}) \tag{3}$$

where $\bar{u(x)}$ is the mean value in a neighborhood of x, and $V(x)$ and V_G represent the variance in the same neighborhood and the global variance of the image, respectively. If the local variance is high (suggesting the presence of structure) the intensity value after filtering will be close to the original pixel value, when the local variance is low (background) the intensity is close to the local mean value.

2 Anisotropic difussion

As described in [1], the image $u(x,t)$ is considered as the solution of the diffusion equation

$$\partial_t u = div(D(|\nabla u|^2)\,|\nabla u|) \tag{4}$$

with initial condition:

$$u(x,0) = g(x).$$

The diffusivity D is often chosen as:

$$D(|\nabla u|^2) = \frac{1}{1+|\nabla u|^2/\lambda^2}$$

which has small values for large gradients, indicating edges. Applied to intracellular image sequences the anisotropic diffusion produces a good visual effect, see fig 4.

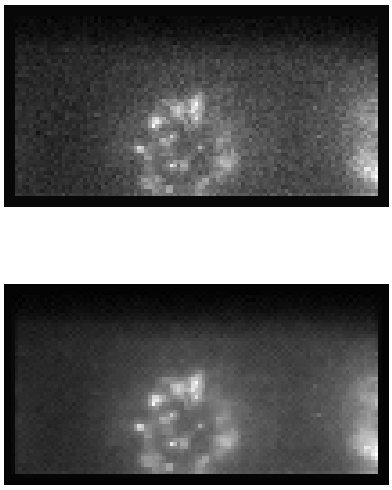

Figure 4: Denoising of a frame in the image sequence by anisotropic diffusion

3 Variational approach

Denoising seen as an inverse problem is ill-posed. where Ω is the image domain, and λ a parameter that determines the trade-off between the two terms. It is proved to accurately estimate discontinuities (edges) of the images.

A maximum likelihood version of ROF adapted for Poisson noise is given in [5]. Considering the pixels x independent, from eq. 1 results that:

$$P(f|u) = \prod_i \frac{e^{-u(x_i)}u(x_i)^{f(x_i)}}{f(x_i)!}. \tag{5}$$

Choosing as regularization term:

$$P(u) := exp\left(-\beta \int_{\Omega} |\nabla u|\right) \tag{6}$$

the problem becomes maximizing the functional $P(f|u)P(u)$. This is equivalent to minimize:

$$-\log\left(P(f|u)P(u)\right) = \sum_{i} (u - f\log u) + \beta \int_{\Omega} |\nabla u| \tag{7}$$

regarded as a discrete approximation of the functional:

$$E(u) := \int_{\Omega} (u - f\log u) + \beta \int_{\Omega} |\nabla u| \,. \tag{8}$$

.

The iterative solution of 8 is given by:

$$u_t := div\left(\frac{\nabla u}{|\nabla u|}\right) + \frac{1}{\beta u}(f - u) \tag{9}$$

with $\frac{\partial u}{\partial \vec{n}} = 0$ on $\partial\Omega$.

Note the similarity to the well-known algorithm due to Rudin, Osher and Fatemi

$$u_t := div\left(\frac{\nabla u}{|\nabla u|}\right) + \lambda(f - u) \tag{10}$$

that minimizes the functional:

$$F(u) := \int_{\Omega} (f - u)^2 + \lambda \int_{\Omega} |\nabla u| \,. \tag{11}$$

The difference consists in the regularization parameter being adapted to the image intesnity.

4 Spatial Tonal Convolution

When the functional to be minimized is chosen as a “soft” least squares:

$$F(u(x)) := \int_{\Omega} \left(f(y) - u(x)\right)^2 G^s(x - y)dy \tag{12}$$

where G^s is a spatial gaussian (2D or 3D), the optimum is achieved for:

$$u_0(x) = \frac{\int_{\Omega} f(y)G^s(x - y)dy}{\int_{\Omega} G^s(x - y)dy} = (f * G)(x) \tag{13}$$

However, if the noise is not Gaussian, (for instance also the case of an image that contains structure) the least square model is not optimal, being very much influenced by outliers (the structure elements). A better approach is offered by robust estimators.

By replacing the least square functional one chooses a robust error measure which is less sensitive to outliers, see [4], the problem becomes the minimization of:

$$E(u) := \int \rho^t \left(f(y) - u(x)\right) G^s(x-y)dy \tag{14}$$

The minimum is achieved when:

$$\int \phi^t \left(f(y) - u(x)\right) G^s(x-y)dy = 0 \tag{15}$$

where ϕ is the derivative of ρ.

By choosing $\rho^t(p) = 1 - exp(-\frac{p^2}{2t^2})$ results that

$$\phi^t(p) = \frac{d\rho^t(p)}{d\rho} = \frac{p}{t^2} exp(-\frac{p^2}{2t^2}) \tag{16}$$

and substituting ϕ^t in 15 we obtain:

$$u(x) = \frac{\int_{R^d} f(y) exp\left(-\frac{(f(y)-u(x))^2}{2t^2}\right) G^s(x-y)dy}{\int_{R^d} exp\left(-\frac{(f(y)-u(x))^2}{2t^2}\right) G^s(x-y)dy} \tag{17}$$

which can be solved using fixed point iteration: $u^{i+1} = F(u^i)$ until convergence or a maximum number of iterations is achieved.

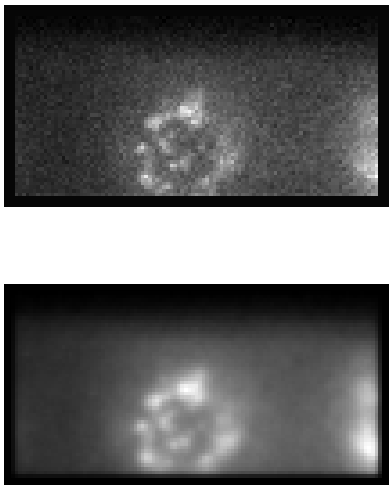

Figure 5: Denoising of a frame in the image sequence with Spatial Tonal Convolution

Remark that the result above is actually the spatial-tonal normalized convolution:

$$[f,u] * [v,w] = \frac{\int_{R^d} f(y) v(x-y) w\left(f(y) - u(x)\right) dy}{\int_{R^d} v(x-y) w\left(f(y) - u(x)\right) dy} \tag{18}$$

where $v = G^s$ and $w = G^{\text{tonal}}$.

5 Conclusions and future work

In this paper, we have presented the 3D version of some popular denoising algorithms for image sequences. The results are promising, although further parameter adjustments might improve the outcome. A comparison of the number of local maxima is being performed (both for background and fore the whole image), in order to evaluate the smoothing effects of the algorithms.

In the future we shall study the incoropration of the optical flow in the denoising process. A possible alternative approach which we shall also consider is represented by steerable filters.

References

[1] Horst Jähne, Bernd Haußecker, *Computer vision and applications*, Academic Press, 2000.

[2] Fionn Murtagh Jean-Luc Starck and Albert Bijaoui, *Image and data analysis: The multiscale approach*, Cambridge University Press, 1998.

[3] Richard E. Woods Rafael C. Gonzalez, *Digital image processing*, Prentice Hall, 2002.

[4] Joost van de Weijer Rein van den Boomgaard, *On the equivalence of local-mode finding, robust estimation and mean-shift*, ICPR (2005).

[5] Thomas J. Asaki Triet Le, Rick Chartrand, *A variational approach to constructing images corrupted by poisson noise*, Tech. report, UCLA CAM Report 05-49, September 2005.

Elicitation of Control Strategies from Data-Based Fuzzy Models

A Tutorial on Control Theory. PhD Seminar November 11, 2005

Rainer Geschray,
V-Research GmbH, Stadtstrasse 33, 6850 Dornbirn, Austria
k0030408@students.jku.at

1 Terms and Definitions

Definition 1.1 ***State Variable.*** *A parameter* $\mathbf{x} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}^T$ *is called a state variable of the system in Fig. 1 with order* $n \in \mathbb{N}$*, if the system behaviour is uniquely specified with known*

(i) initial state $\mathbf{x}_0 = \mathbf{x}(t_0)$*, where* $t = t_0$ *is an arbitrary point in time and*

(ii) some system input $\mathbf{u}(t) = \begin{bmatrix} u_1 & \dots & u_m \end{bmatrix}^T$*,* $m \in \mathbb{N}$ *and* $t \geq t_0$*.*

The system behaviour is characterised by

$$\mathbf{x}(t) = \Gamma \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{u}(t) \end{pmatrix}. \tag{1}$$

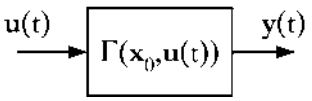

Figure 1: System Structure

Definition 1.2 ***Time Continuous and Discrete Systems.*** *If the time domain of the system (1) is*

$$t \in \{x | t_0 \leq x \leq t_1; x, t_0, t_1 \in \mathbb{R}\} \tag{2}$$

the system is termed ***time continuous****. If the time domain is*

$$t \in \{x | t_0 \leq x \leq t_1, x = nt_d; t_0, t_1, t_d \in \mathbb{R}, n \in \mathbb{N}\} \tag{3}$$

the system is termed ***time discrete****.*

Definition 1.3 ***Linearity.*** *The system*

$$\mathbf{x}(t) = \Gamma \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{u} \end{pmatrix} \tag{4}$$

is called ***linear****, if the principle of superposition*

$$\Gamma \left(\alpha \begin{bmatrix} \mathbf{x}_{01} \\ \mathbf{u}_1 \end{bmatrix} + \beta \begin{bmatrix} \mathbf{x}_{02} \\ \mathbf{u}_2 \end{bmatrix} \right) = \alpha \Gamma \begin{pmatrix} \mathbf{x}_{01} \\ \mathbf{u}_1 \end{pmatrix} + \beta \Gamma \begin{pmatrix} \mathbf{x}_{02} \\ \mathbf{u}_2 \end{pmatrix} \tag{5}$$

holds with the arbitrary constants α*,* β*,* $\mathbf{x}_{01}$*,* $\mathbf{x}_{02}$ *and functions* $\mathbf{u}_1$*,* $\mathbf{u}_2$*. If (5) does not hold, the system is* ***nonlinear****.*

Definition 1.4 ***Time Variance.*** *The system (1) is called* ***time invariant****, if from*

$$\mathbf{x}(t) = \Gamma \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{u}(\tau) \end{pmatrix} \quad t_0 \leq \tau \leq t \tag{6}$$

followes with the shifted initial time point $(t_0 + T)$

$$\mathbf{x}(t - T) = \Gamma \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{u}(\tau - T) \end{pmatrix}. \tag{7}$$

This concludes that the state vector $\mathbf{x}$ *is a function of the time difference* $(t - t_0)$*, i.e.* t_0 *can be set to zero without loss of generality.*

1.1 Stability

Stability is a fundamental property of the closed loop control system. An unstable control system is more or less unusable, i.e. stability is a major constraint in controller design!

1.1.1 Linear Systems

In this section the focus lies on linear systems according definition 1.3. In the next paragraph, synonymous descriptions of linear systems are illustrated followed by common stability terms of linear systems.

Description of Linear Time Invariant Continuous Systems without Time delay in State Space. Time continuous systems of order n, which is linear in its state $\mathbf{x} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}^T$ and input variable $\mathbf{u} = \begin{bmatrix} u_1 & \dots & u_m \end{bmatrix}^T$, can be described with

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{8}$$

where $\dot{\mathbf{x}} = \dfrac{d\mathbf{x}}{dt}$ is the derivation of time and

$$\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} b_{11} & \dots & b_{1m} \\ \vdots & \ddots & \vdots \\ b_{n1} & \dots & b_{nm} \end{bmatrix} \tag{9}$$

are constant matrices. The system output $\mathbf{y} = \begin{bmatrix} y_1 & \dots & y_r \end{bmatrix}^T$ is a linear combination of the state variables and the system input

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \tag{10}$$

where

$$\mathbf{C} = \begin{bmatrix} c_{11} & \dots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{r1} & \dots & c_{rn} \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} d_{11} & \dots & d_{1m} \\ \vdots & \ddots & \vdots \\ d_{r1} & \dots & d_{rm} \end{bmatrix} \tag{11}$$

are constant matrices. In practical applications, where the plant has a lowpass characteristic, $\mathbf{D} = \mathbf{0}$ becomes the Nullmatrix. The solution of (8) is

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}_0 + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau, \tag{12}$$

where $\mathbf{x}_0 = \mathbf{x}(t=0)$ and $e^{\mathbf{A}t} = \sum_{\nu=0}^{\infty} \dfrac{\mathbf{A}^\nu t^\nu}{\nu!}$.

Example 1.1 *RLC circuit of order $n = 2$, Figure 2. The choice of the state variable $\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$ is **not unique**! In this example, x_1 is the voltage of the capacitor C_1 and x_2 is the current through the inductance L_1, but there is an infinity number of other opportunities to choose x_1 and x_2. If the voltage of the inductance L_1 is the output y of the circuit, the system is*

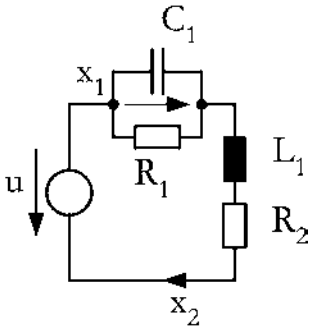

Figure 2: Example - RLC circuit

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\dfrac{1}{R_1 C_1} & \dfrac{1}{C_1} \\ -\dfrac{1}{L_1} & -\dfrac{R_2}{L_1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{L_1} \end{bmatrix} u \tag{13}$$

$$y = \begin{bmatrix} -1 & -R_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix} u. \tag{14}$$

With

$$C_1 = 2, \quad L_1 = 4, \quad R_1 = 1, \quad R_2 = 3$$

(13,14) becomes

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -0.5 & 0.5 \\ -0.25 & -0.75 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.25 \end{bmatrix} u \tag{15}$$

$$y = \begin{bmatrix} -1 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + u. \tag{16}$$

Description of Linear Time Invariant Continuous Systems in Frequency Space. With the Laplace transform

$$f(s) = \mathcal{L}\{f(t)\} = \int_0^\infty f(t)e^{-st}dt, \tag{17}$$

which is convergent in $Re(s) > K, \quad K \in \mathbb{R}$. The *transfer function* $G_{ij}(s)$ is the ratio of the Laplace transformed output $y_i(s) = \mathcal{L}\{y_i(t)\} \quad i = 1 \dots r$ and input $u_j(s) = \mathcal{L}\{u_j(t)\} \quad j = 1 \dots m$ of the

system (8,10) with zero initial state $\mathbf{x}_0 = \mathbf{0}$.

$$G_{ij}(s) := \left.\frac{y_i(s)}{u_j(s)}\right|_{\mathbf{x}_0=\mathbf{0}} \quad (18)$$

$$\mathbf{G}(s) = \begin{bmatrix} G_{11}(s) & \dots & G_{1m}(s) \\ \vdots & \ddots & \vdots \\ G_{r1}(s) & \dots & G_{rm}(s) \end{bmatrix} = \mathbf{C}\left(s\mathbf{I}-\mathbf{A}\right)^{-1}\mathbf{B}+\mathbf{D} \quad (19)$$

where $\mathbf{I}$ is the identity matrix.

Definition 1.5 ***Steady State*** $\mathbf{x}_R$. *In a steady state of the system (8) the derivation $\dot{\mathbf{x}} = \mathbf{0}$ vanishes. With $\mathbf{u} = \mathbf{0}$, (8) becomes $\mathbf{A}\mathbf{x}_R = \mathbf{0}$, i.e. a linear time invariant system has always $\mathbf{x}_R = \mathbf{0}$ and no other steady state if $\mathbf{A}$ is regular. If $\mathbf{A}$ is singular there are infinity steady states. This means the eigendirection $\mathbf{x}_R$ with corresponding eigenvalue 0 in $\mathbf{A}\mathbf{x}_R = 0 \cdot \mathbf{x}_R$).*

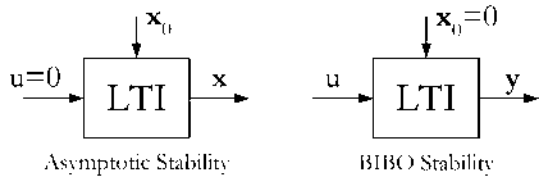

Figure 3: Two System Experiments

Definition 1.6 ***Bounded Input Bounded Output BIBO Stability.*** *If the input of the right system in Fig. 3 is bounded with*

$$|\mathbf{u}(t)| \le u_{max} < \infty \quad \forall t,$$

the linear time invariant continuous system ($r = 1$, $m = 1$)

$$y(t) = \int_0^\infty g(t-\tau)u(\tau)d\tau + Du(t) \quad (20)$$

with

$$g(t) = \mathbf{C}e^{\mathbf{A}t}\mathbf{B} \quad (21)$$

is Bounded Input Bounded Output BIBO stable, i.e.

$$|\mathbf{y}(t)| \le y_{max} < \infty \quad \forall t,$$

iff

$$\int_0^\infty |g(\tau)|d\tau < \infty. \quad (22)$$

For systems (21) it's a necessary and sufficient criterion for BIBO stability, that the transfer function $\mathbf{G}(s) = \mathbf{C}\left(s\mathbf{I}-\mathbf{A}\right)^{-1}\mathbf{B}+\mathbf{D}$ has no poles in $\mathbb{C}_{+e}$ (i.e. the secluded right complex halfplane extended with the point $s = \infty$). This is equivalent to the criterion:

* $\Re\{poles(\mathbf{G}(s))\} < 0$ *and*
* $\lim_{s\to\infty} |G_{ij}(s)| < \infty \quad i = 1 \dots r, j = 1 \dots m$

Example 1.2 ***Transfer function*** $G(s)$ ***of RLC circuit in Example 1.1.*** *The poles of the transfer function*

$$\mathbf{G}(s) = \mathbf{C}\left(s\mathbf{I}-\mathbf{A}\right)^{-1}\mathbf{B}+\mathbf{D} = \begin{bmatrix}-1 & -3\end{bmatrix}\begin{bmatrix} s+0.5 & -0.5 \\ 0.25 & s+0.75\end{bmatrix}^{-1}\begin{bmatrix}0 \\ 0.25\end{bmatrix}+1 = \frac{s^2+0.5s}{s^2+1.25s+0.5}$$

are

$$s_{1,2} = -0.625 \pm j0.33072$$

and

$$\lim_{s\to\infty} |G(s)| = 1,$$

i.e. the system is BIBO stable.

Definition 1.7 ***Asymptotic Stability.*** *The left system in Fig. 3*

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} \quad (23)$$

is called asymptotic stable, if the state $\mathbf{x}$ ends allways in $\lim_{t\to\infty}\mathbf{x}(t) = \mathbf{0} \quad \forall \mathbf{x}_0$. A necessary and sufficient criterion for asymptotic stability is

$$\Re\left\{s_i(\mathbf{A})\right\} < 0 \quad i = 1 \dots n, \quad (24)$$

$\mathbf{A}$ is then called a Hurwitz matrix. The system (23) is unstable, if there is at least one initial state $\mathbf{x}_0$, such that $\lim_{t\to\infty} x_i(t) = \infty \quad i = 1 \vee 2 \vee \dots n$. A sufficient criterion for instability is that there is at leas one eigenvalue of $\mathbf{A}$ with a real part greater

than zero. If all eigenvalues differ from each other, i.e.

$$s_i \neq s_j \quad \forall i,j = 1 \ldots n, \quad i \neq j$$

the criterion is also necessary. If there are some equal eigenvalues, instable behaviour ist possible although there are no eigenvalues with real part greater than zero (e.g. double integrator).

1.1.2 Nonlinear Systems

The system (1) is called nonlinear, if (1.3) doesn't hold. There is an essential distinction in the stability concept of linear time invariant and nonlinear systems!

LTI system Stability is a system feature.

NL system There is any number of steady states possible and each of them has its own stability characteristic.

Definition 1.8 ***Lyapunov Stability.*** *The steady state* $\mathbf{x}_R$ *of the system*

$$\dot{\mathbf{x}} = \Gamma(\mathbf{x}) \tag{25}$$

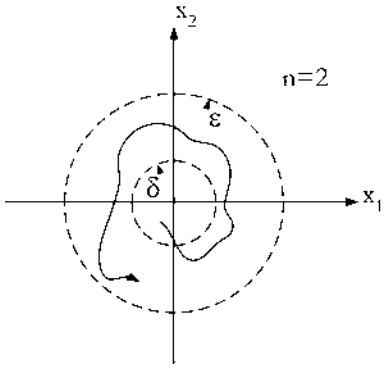

Figure 4: Lyapunov Stability for a System with Order $n = 2$. Coordinate origin is $\mathbf{x}_R$.

is called (Lyapunov) stable, if the trajectory $\mathbf{x}(t)$ *stays for* $t \geq t_0$ *in the area of* ε*, when the initial state* $\mathbf{x}(t_0)$ *is in the area of* δ*, i.e.*

$$\mathbf{x} = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix}^T$$

$$||\mathbf{x}||_p = \left(\sum_{\nu=1}^{n} |x_\nu|^p \right)^{\frac{1}{p}} \quad p \in \mathbb{N}$$

$$||\mathbf{x}|| \Rightarrow p = 1 \ldots \infty$$

$$\forall \varepsilon > 0 \quad \exists \delta > 0 \quad ||\mathbf{x}(t_0)|| < \delta \quad \Rightarrow \quad ||\mathbf{x}(t)|| < \varepsilon \quad t \geq t_0.$$

Definition 1.9 ***Asymptotic Stability.*** *The steady state* $\mathbf{x}_R$ *of the system (25) is called asymptotic stable, if* $\mathbf{x}_R$ *is (Lyapunov) stable and the trajectory* $\mathbf{x}(t)$*, who starts in a sufficient small area around* $\mathbf{x}_R$*, move towards* $\mathbf{x}_R$ *for* $t \to \infty$*.*

Other Stability Terms: Absolute Stability, Superstability

2 Closed Loop Position-Control of an Iron Ball

The general controller design is divided into

(i) Listing of requirements, specifications and constraints

(ii) Plant modelling, e.g. differential/algebraic equations, data based models

(iii) Set up an adequate control structure

(iv) Controller design

- Standard controller, e.g. PID controller
- Methods in frequency space, e.g. algebraic synthesis, frequency characteristic curve
- Methods in state space, e.g. state observer and controller
- Controller design for plants with significant delay time, e.g. Smith-Predictor

This section illustrates the controller design for the nonlinear plant in Fig. 5.

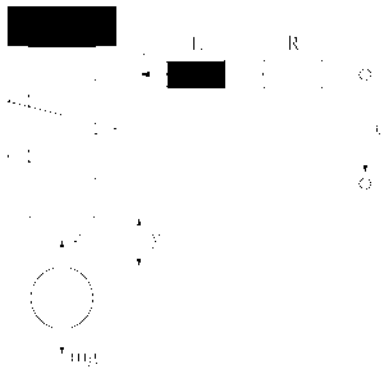

Figure 5: The Iron Ball Plant

2.1 Specifications

* The system consists of an coil and an iron ball.
* The position y of the iron ball is the target variable for control.
* Input variable is the voltage u of the coil. R and L describe the characteristic of the coil.
* Stationary control error $\lim_{t\to\infty} e(t) = e_\infty = y_{guide} - y_\infty = 0$, i.e. the control error e_∞ vanishes in the case of a constant command variable Δr_k.

2.2 Plant Equations

With the state vector

$$\mathbf{x} := \begin{bmatrix} y & \frac{dy}{dt} & i \end{bmatrix}^T$$

the nonlinear model

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u)$$
$$y = g(\mathbf{x}, \mathbf{u})$$

becomes

$$\begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \\ \dot{\mathbf{x}}_3 \end{bmatrix} = \begin{bmatrix} x_2 \\ g - \frac{cx_3^2}{mx_1^2} \\ -\frac{R}{L}x_3 + \frac{2cx_2x_3}{Lx_1^2} + \frac{1}{L}u \end{bmatrix}$$
$$y = x_1.$$

There is one steady state ($\dot{\mathbf{x}} = \mathbf{0}$)

$$\mathbf{x}_R = \begin{bmatrix} y_R & 0 & y_R\sqrt{\frac{mg}{c}} \end{bmatrix}^T$$
$$u_R = Ry_R\sqrt{\frac{mg}{c}}.$$

2.3 Control Strategy

(i) Linearization around the steady state of the nonlinear model.

(ii) Time discretization of the linear model.

(iii) Design of an Luenberger observer.

(iv) Design of an state controller (Fig. 6).

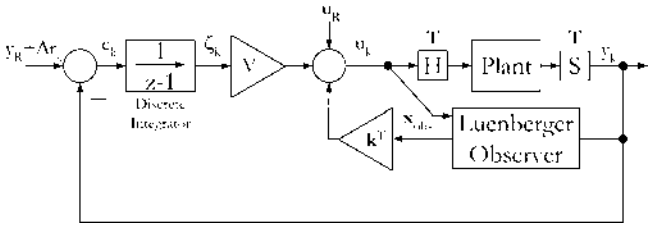

Figure 6: The State Controller with the Luenberger Observer

2.3.1 Linearization around $\mathbf{x}_R$

With the transformation

$$\Delta\mathbf{x} := \mathbf{x} - \mathbf{x}_R$$
$$\Delta u := u - u_R$$

the linearized model around the steady state is

$$\dot{\mathbf{x}} = \Delta\dot{\mathbf{x}} = \left.\frac{\partial \mathbf{f}(\mathbf{x}, u)}{\partial \mathbf{x}}\right|_{\mathbf{x}_R, u_R} \Delta\mathbf{x} + \left.\frac{\partial \mathbf{f}(\mathbf{x}, u)}{\partial u}\right|_{\mathbf{x}_R, u_R} \Delta u$$
$$\Delta y = y - y_R = \left.\frac{\partial g(\mathbf{x}, u)}{\partial \mathbf{x}}\right|_{\mathbf{x}_R, u_R} \Delta\mathbf{x} + \left.\frac{\partial g(\mathbf{x}, u)}{\partial u}\right|_{\mathbf{x}_R, u_R} \Delta u$$

$$\mathbf{A} := \left.\frac{\partial \mathbf{f}(\mathbf{x},u)}{\partial \mathbf{x}}\right|_{\mathbf{x}_R,u_R} \quad \mathbf{B} := \left.\frac{\partial \mathbf{f}(\mathbf{x},u)}{\partial u}\right|_{\mathbf{x}_R,u_R}$$
$$\mathbf{C} := \left.\frac{\partial g(\mathbf{x},u)}{\partial \mathbf{x}}\right|_{\mathbf{x}_R,u_R} \quad \mathbf{D} := \left.\frac{\partial g(\mathbf{x},u)}{\partial u}\right|_{\mathbf{x}_R,u_R}$$

$$\Delta\dot{\mathbf{x}} = \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\Delta u \tag{26}$$
$$\Delta y = \mathbf{C}\Delta\mathbf{x} + \mathbf{D}\Delta u \tag{27}$$
$$g = 9.81\frac{m}{s^2}$$
$$c = 136.32 \cdot 10^{-6}\frac{kgm^3}{s^2A^2}$$
$$m = 66.87g$$
$$L = 1.08H$$
$$R = 18\Omega$$
$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{2g}{y_R} & 0 & -\frac{2}{y_R}\sqrt{\frac{cg}{m}} \\ 0 & \frac{2}{Ly_R}\sqrt{cmg} & -\frac{R}{L} \end{bmatrix}$$
$$\mathbf{B} = \begin{bmatrix} 0 & 0 & \frac{1}{L} \end{bmatrix}^T$$
$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$
$$\mathbf{D} = 0.$$

2.3.2 Discrete Time Linearized Model

If the input variable $u(t)$ consists of piecewise polynomes of zero-order (stairs), the system (26,27) becomes

$$\Delta\mathbf{x}_{k+1} = \mathbf{A}_D\Delta\mathbf{x}_k + \mathbf{B}_D\Delta u_k \tag{28}$$
$$\Delta y_k = \mathbf{C}_D\Delta\mathbf{x}_k + \mathbf{D}_D\Delta u_k, \tag{29}$$

where $T = 5ms$ is the Time Constant and

$$\mathbf{A}_D = e^{\mathbf{A}T}$$
$$\mathbf{B}_D = \int_0^T e^{\mathbf{A}\tau}\mathbf{B}d\tau$$
$$\mathbf{C}_D = \mathbf{C}$$
$$\mathbf{D}_D = \mathbf{D}.$$

(28,29) is characterised with $[\mathbf{A}_D, \mathbf{B}_D, \mathbf{C}_D, \mathbf{D}_D]$.

2.3.3 Luenberger State Observer

The state observer $\Delta\mathbf{x}_{obs,k}$ has the form

$$\begin{aligned}\Delta\mathbf{x}_{obs,k+1} &= \mathbf{A}_D\Delta\mathbf{x}_{obs,k} + \mathbf{B}_D\Delta u_k + \mathbf{f}\left(\Delta y_{obs,k} - \Delta y_k\right) \\ &= \left(\mathbf{A}_D + \mathbf{f}\mathbf{C}_D\right)\Delta\mathbf{x}_{obs,k} + \\ &\quad + \mathbf{B}_D\Delta u_k - \mathbf{f}\Delta y_k,\end{aligned}$$

where $\mathbf{f}$ is chosen to meet eigenvalues of $(\mathbf{A}_D + \mathbf{f}\mathbf{C}_D)$ within the unit circle of the z-plane.

2.3.4 State Controller with Discrete Integrator

The z-Transform of the causal sequence $(f) = (f_0, f_1, f_2, \dots)$ is

$$f(z) := \sum_{i=0}^{\infty} f_i z^{-i}, \tag{30}$$

which is convergent in the area $|z| > K \in \mathbb{R}$. If the limiting value of the control error exists (when command variable Δr_k has a constant value), it can be acquired with the theorem

$$e_\infty := \lim_{k\to\infty} e_k = \lim_{z\to 1}(z-1)e(z).$$

Then, the steady state error becomes

$$\begin{aligned} e_\infty &= \lim_{z\to 1}(z-1)e(z) \\ &= \lim_{z\to 1}(z-1)E(z)\frac{z}{z-1} \\ &= \lim_{z\to 1} zE(z) \\ &= \lim_{z\to 1} z\frac{1}{1+P(z)R(z)} \\ &= \lim_{z\to 1} z\frac{z-1}{z+P(z)\tilde{R}(z)} = 1\frac{1-1}{1+P(1)\tilde{R}(1)} = 0 \end{aligned}$$

$$P(1)\tilde{R}(1) \neq -1.$$

$$f(z) = \frac{z}{z-1} \tag{31}$$

is the z-transformed unit step $(f) = (1, 1, \dots)$. $E(z) = e(z)/r(z)$ is the ratio of the z-transformed command variable r_k and error variable e_k in the standard closed loop control in Fig. 7. I.e., if there is an integral behaviour of the controller $R(z)$, the

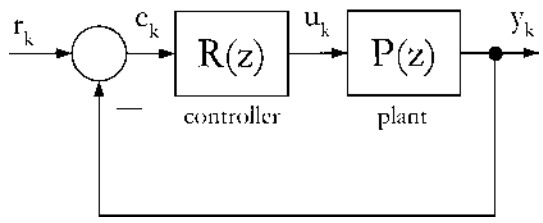

Figure 7: Standard closed loop control

steady state error e_∞ becomes zero! The state controller is

$$\Delta u_k = \mathbf{K}\Delta\mathbf{x}_k + V\zeta_k,$$

where $\mathbf{K}$ and V are constants. With (28) the overall system (plant and state controller) is

$$\Delta\mathbf{x}_{k+1} = (\mathbf{A}_D + \mathbf{B}_D\mathbf{K})\,\Delta\mathbf{x}_k + \mathbf{B}_D V\zeta_k.$$

The discrete time integrator is

$$\begin{aligned}\zeta_{k+1} &= \zeta_k + e_k = \zeta_k + \Delta r_k - \Delta y_k \\ &= \zeta_k - \mathbf{C}_D\Delta\mathbf{x}_k.\end{aligned}$$

The system of plant, state controller and integrator is ($\Delta r_k = (0)$)

$$\begin{bmatrix}\Delta\mathbf{x}_{k+1}\\ \zeta_{k+1}\end{bmatrix} = \begin{bmatrix}\mathbf{A}_D + \mathbf{B}_D\mathbf{K} & \mathbf{B}_D V\\ -\mathbf{C}_D & 1\end{bmatrix}\begin{bmatrix}\Delta\mathbf{x}_k\\ \zeta_k\end{bmatrix},$$

i.e. the state controller is designed by meeting the eigenvalues of (2.3.4) within the unit circle $|z| < 1$.

2.4 Experiments

Fig. 8 mapps the output sequence y_k, if the command variable r_k is $(r) = (15, 15, \dots)$.

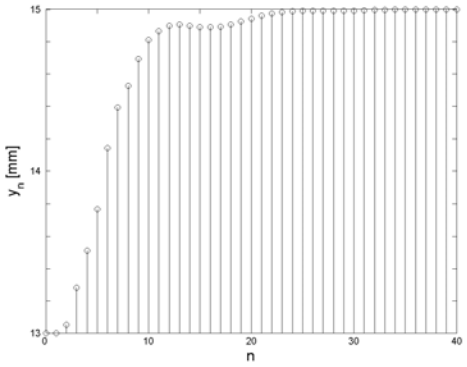

Figure 8: Step Response with $dy = 2mm$

References

[1] Horn M., *Computer Aided System Theory*, TU Graz, November 2003.

[2] Horn M., *Nonlinear Control Systems*, TU Graz, March 2003

[3] Hofer A., *Design of Optimal Systems*, TU Graz, April 2004

[4] Dourdoumas N. & Horn M., *Regelungstechnik*, Pearson Studium, Graz 2004

Technical Report
SCCH–TR–0534

Segmentation and Tracking by Mean Shift
A Tutorial

Bernhard Moser
Software Competence Center Hagenberg
A-4232 Hagenberg, Austria
bernhard.moser@scch.at

Abstract — The mean shift algorithm is a method for estimating the mode of a density function. It is demonstrated how this method can be used for image segmentation preserving discontinuities and for tracking purposes.

Key words — *Segmentation, Tracking, Adaptive Gradient Method, Density Approximation*

Kplus
Kompetenzzentren-Programm

1 Concept of Mean Shift, Feature Space Analysis

The mean shift algorithm is a method for estimating the mode of a density function, see [1, 2, 4]. In this tutorial we follow mainly the ideas as presented in the papers [2] and [3]. Given identically distributed data points $x_1, \ldots, x_n \in \mathcal{X} = \mathbb{R}^d$, find an approximation of the underlying distribution, at least the mode of it!

To approximate the unerlying distribution thh following approach is chosen

$$\hat{f}(x) = \frac{1}{n} \sum_i K_H(x - x_i)$$

where

- H symmetric and positive definite, $K : \mathcal{X} \to [0, \infty)$:

$$\int_{\mathcal{X}} K(x)dx = 1, \quad \lim_{\|x\|\to\infty} \|x\|^d K(x) = 0$$
$$\int_{\mathcal{X}} xK(x)dx = 0, \quad \int_{\mathcal{X}} xx^T K(x)dx = c_K I$$

- $K_H(x) = |H|^{-\frac{1}{2}} K(H^{-\frac{1}{2}}x)$

For simplicity often only a single parameter bandwidth matrix H is used:

$$H = h^2 I \implies |H|^{-\frac{1}{2}} = \frac{1}{h^d},$$

hence

$$\hat{f}_{h,K}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{1}$$

For simplicity let us generate the kernel windowing function $K(.)$ by a profile due to

$$K(x) = k(\|x\|^2).$$

By this we obtain an estimation for the density gradient:

$$\begin{aligned} \nabla \hat{f}_{h,K}(x) &= \frac{c}{h^{d+2}} \cdot \sum_i (x_i - x) g(\xi_i) \\ &= \frac{c}{h^{d+2}} \cdot \underbrace{\left(\frac{\sum_i x_i g(\xi_i)}{\sum_i g(\xi_i)} - x \right)}_{\text{mean shift } m_{h,G}} \cdot \sum_i g(\xi_i) \end{aligned} \tag{2}$$

where $g(.) = -k'(.)$ and $\xi_i = \|\frac{x - x_i}{h}\|^2$.

Formula (2) shows that the mean shift algorithm is actually an adaptive gradient ascent method

$$m_{h,G}(x) = c \cdot h^2 \frac{\nabla \hat{f}_{h,K}(x)}{\hat{f}_{h,G}(x)}$$

Note that for high density estimates we get small steps and for low density ones we get bigger steps.

The most often used profiles are

- **Gaussian**

$$k_G(x) = \begin{cases} e^{-\frac{1}{2}x} & \text{if } x > 0, \\ 0 & \text{else} \end{cases}$$

- **Epanechnikov**

$$k_E(x) = \begin{cases} 1 - x & \text{if } 0 \leq x \leq 1, \\ 0 & x > 1 \end{cases}$$

2 Algorithm and Convergence

The following theorem can be found in [2].

Theorem 1. *Let the kernel K have a convex and monotonically decreasing profile, then*

$$\exists \lim_{i \to \infty} \tilde{x}_i$$
$$\exists \lim_{i \to \infty} \hat{f}_{h,K}(\tilde{x}_i)$$

where $\tilde{x}_{i+1} = m_{h,G}(\tilde{x}_i)$. $\left(\hat{f}_{h,K}(\tilde{x}_i)\right)_i$ is monotonically increasing.

Figure 2 shows a set of points generated according to a Gaussian distribution and Epanechnikov estimates by formula 1 for different bandwidth values 0.2, 0.4, 0.6 and 0.8. It can be oberserved that lower the bandwidth the estimate is the more sensitive to local densities. The blue points depict a sequence of iterations generated by the mean shift for bandwidth 0.2 showing that the corresponding estimate values are monotonically increasing.

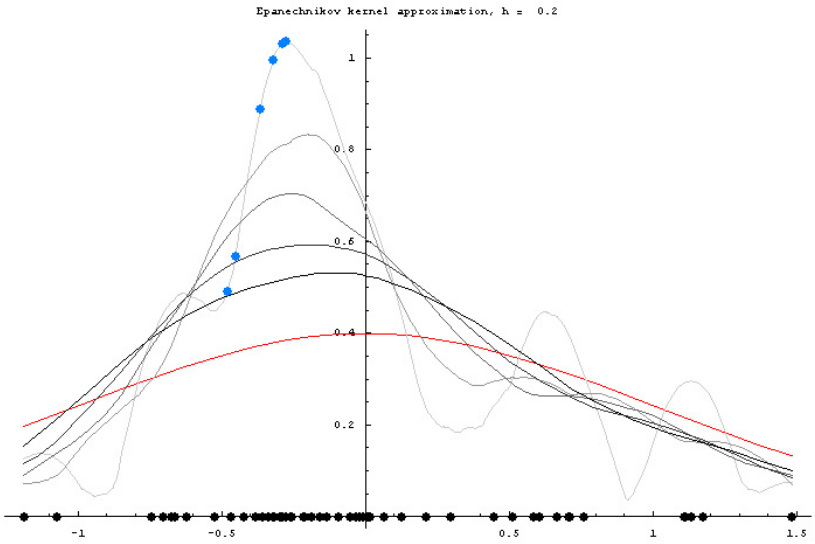

Figure 1: Illustration of Epanechnikov estimates (grey curves) of a Gaussian randomly distribution (red curve). The dots depict a convergent sequence generated by the mean shift algorithm.

3 Application to Images

Theorem 1 can be slightly extended for positive weights in the following way: For

$$\begin{aligned} \hat{f}(x) &= \frac{\sum_i \alpha_i K_H(x-x_i)}{\sum_i \alpha_i} \\ m_{h,G} &= \frac{\sum_i x_i \alpha_i g(\xi_i)}{\sum_i \alpha_i g(\xi_i)} - x \end{aligned}$$

where $\alpha_i \geq 0$ and $\xi_i = \|\frac{x-x_i}{h}\|^2$ the convergence theorem 1 still holds true.

By choosing the weights α_i to be the intensity levels of the pixels of an image we obtain a formular which in this way can be applied to images.

3.1 Application I: Segmentation

For instance the basins of attraction provide a segmentation which is discontinuity preserving. This is effect is illustrated by figure 3.1.

Figure 2: Demonstration of discontinuity preserving segmentation by mean shift (left: orignal, right: result)

3.2 Application I: Tracking

For tracking pruposes we can proceed as follows:

- start with a characterizing histogram of the target.
- calculate backprojection $\alpha_i = I(x_i, y_i)$, where $I(x, y)$ denotes the intensity level of the pixel (x, y)
- define starting window W

- apply mean-shift to backprojection, e.g., the simplest form (Epanechnikov)

$$\text{new center} = \frac{1}{\sum_{(x,y)\in W} I(x,y)} \begin{pmatrix} \sum_{(x,y)\in W} xI(x,y) \\ \sum_{(x,y)\in W} yI(x,y) \end{pmatrix}$$

This is illustrated by the figures 3. The left shows the tracking window W, the right its hue-histogram and, finally, figure 4 the resulting backprojection.

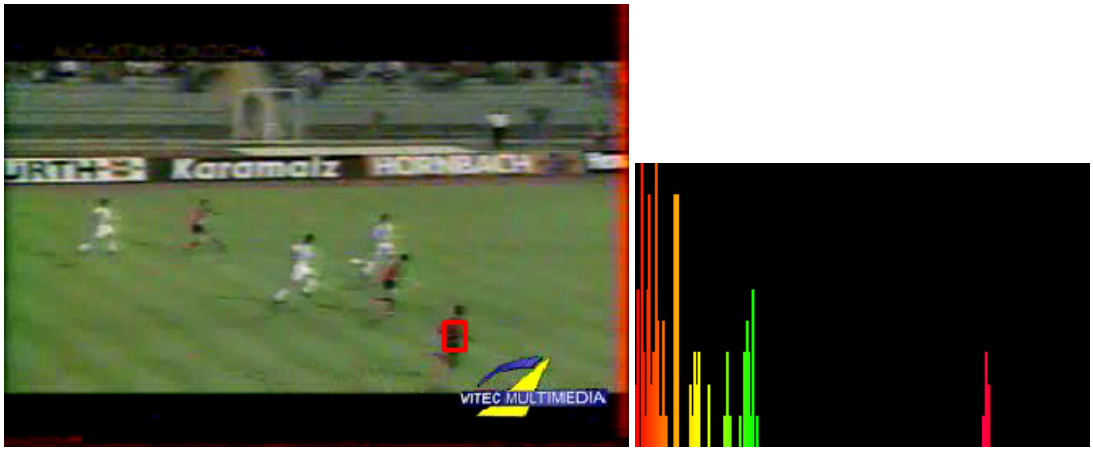

Figure 3: tracking window and histogram

Figure 4: example of backproject

If the features and histogram is properly chosen there is only a small number (1 till 3) of iterations needed let the mean shift algorithm converge. Therefore, the mean shift is a proper tool for tracking purposes in real time.

References

[1] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[2] D. Comaniciu and P. Meer. Robust analysis of feature spaces: color image segmentation. In *Proc. of the 1997 Conf. on Computer Vision and Pattern Recognition (CVPR 97)*, Washington, DC, 1997. IEEE Computer Society.

[3] V. Ramesh D. Comaniciu and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 142–149, Hilton Head, SC, 2000.

[4] K. Fukunaga and L.D. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Information Theory*, 21:32–40, 1975.

COPULAS: NEW RESULTS AND OPEN PROBLEMS

RADKO MESIAR

Department of Mathematics and Descriptive Geometry
Faculty of Civil Engineering
Slovak University of Technology
SK-813 68 Bratislava, Slovakia

email: mesiar@math.sk

Abstract

Some recent results in the field of copulas are presented. Several open problems are included.

1 Introduction

Copulas were introduced by Sklar [26] to describe the dependence structure of random vectors. For a two dimensional case, a copula $C : [0,1]^2 \to [0,1]$ is an aggregation operator with neutral element 1, which is 2–increasing, i.e.,

$$C(x,y) + C(x',y') - C(x,y') - C(x',y) \geq 0 \tag{1}$$

for all $x \leq x'$, $y \leq y'$.
Similarly, copulas with higher dimension n can be introduced, replacing 2–increasingness (1) by the n–increasingness:

$$\sum_{\epsilon_i \in \{-1,1\}^n} (-1)^{\prod_{i=1}^{n} \epsilon_i} C(x_1^{(\epsilon_1)}, \ldots, x_n^{(\epsilon_n)}) \geq 0, \tag{2}$$

for all $0 \leq x_1^{(-1)} \leq x_1^{(1)} \leq 1, \ldots, x_n^{(-1)} \leq x_n^{(1)} \leq 1$. As an example recall the copula $\Pi : [0,1]^n \to [0,1]$, $\Pi(x_1, \ldots, x_n) = x_1 \ldots . x_n$ which is an n–copula for each $n \geq 2$. When speaking about copulas without specification of their dimension, we will always have in mind 2–copulas, i.e., 2–dimensional case.

According to Sklar's theorem, for any random vector $H = (X,Y)$ there is a copula C such that

$$F_H(u,v) = C\left(F_X(u), F_Y(v)\right) \quad \text{for all } u,\, v \in \overline{\mathbb{R}},$$

where F_H, F_X, F_Y are the distribution functions of H, X, Y, respectively. Moreover, C is determined uniquely on $\operatorname{Ran} F_X \times \operatorname{Ran} F_Y$ (more precisely, on $\overline{\operatorname{Ran} F_X} \times \overline{\operatorname{Ran} F_Y}$), and the restriction $C|_{(\operatorname{Ran} F_X \times \operatorname{Ran} F_Y)}$ is called a subcopula. Vice–versa, for any closed subsets A, B of $[0,1]$ containing 0 and 1, a mapping $D : A \times B \to [0,1]$ which is 2–increasing, 1 is its neutral element and which is also non–decreasing, is always a subcopula of some copula C. Evidently, if $H = (X,Y)$ is a discrete random vector, the corresponding subcopula, which is then unique, is defined on a discrete set. For more details about copulas we recommend Nelsen's book [22] and the monograph [25].

Suppose that random variables X and Y are coupled by a copula C. Then for any increasing $\mathbb{R} \to \mathbb{R}$ transformations f_1, f_2 and any decreasing $\mathbb{R} \to \mathbb{R}$ transformations g_1, g_2, random

variables $f_1(X)$ and $f_2(Y)$ are also coupled by C, but random variables $f_1(X)$ and $g_2(Y)$ are coupled by a copula $C^- : [0,1]^2 \to [0,1]$ given by

$$C^-(x,y) = x - C(x, 1-y), \tag{3}$$

and similarly, random variables $g_1(X)$ and $f_2(Y)$ are coupled be a copula $C_- : [0,1]^2 \to [0,1]$ given by

$$C_-(x,y) = y - C(1-x, y). \tag{4}$$

Copulas $g_1(X)$ and $g_2(Y)$ are coupled by the survival copula $\hat{C} : [0,1]^2 \to [0,1]$ given by

$$\hat{C}(x,y) = x + y - 1 + C(1-x, 1-y). \tag{5}$$

Observe that

$$(C^-)^- = (C_-)_- = \widehat{(\hat{C})} = C \quad \text{and} \quad (C^-)_- = (C_-)^- = \hat{C}.$$

Several interesting results concerning these related copulas can be found in [8]. Note that constructions (3), (4), (5) allow to extend or to modify several results for copulas. As a typical example recall W–ordinal sums introduced in [18]. This new type of ordinal sums for copulas can be derived from standard ordinal sums of copulas (we will call them M–ordinal sums) by means of either (3) or (4). Indeed, $C = W(\langle a_\alpha, b_\alpha, C_\alpha \rangle \,|\, \alpha \in A)$ is given by

$$C(x,y) = \begin{cases} (b_\alpha - a_\alpha)\, C\left(\frac{x-a_\alpha}{b_\alpha - a_\alpha}, \frac{y-(1-a_\alpha)}{b_\alpha - a_\alpha}\right) & \text{if } (x,y) \in [a_\alpha, b_\alpha] \times [1-b_\alpha, 1-a_\alpha] \\ W(x,y) & \text{else,} \end{cases}$$

where $W(x,y) = \max(x+y-1, 0)$, if and only if

$$C^- = M(\langle a_\alpha, b_\alpha, C_\alpha^- \rangle \,|\, \alpha \in A),$$

i.e.,

$$C^-(x,y) = \begin{cases} a_\alpha + (b_\alpha - a_\alpha)\, C_\alpha^-\left(\frac{x-a_\alpha}{b_\alpha - a_\alpha}, \frac{y-a_\alpha}{b_\alpha - a_\alpha}\right) & \text{if } (x,y) \in [a_\alpha, b_\alpha]^2 \\ M(x,y) & \text{else,} \end{cases}$$

where $M(x,y) = \min(x,y)$.

2 Discrete copulas

An interesting class of copulas are discrete copulas introduced in [14], compare also empirical copulas discussed in [22], $D : I_n^2 \to [0,1]$ (or in m–dimensional case, $D : I_n^m \to [0,1]$), where $I_n = \{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}$, and irreducible discrete copulas $K : I_n^2 \to I_n$ ($K : I_n^m \to I_n$) introduced in [16] (in an equivalent form on the scales $L_n = \{0, 1, \ldots, n\}$), see also [17].

Discrete copulas $D : I_n^2 \to [0,1]$ are in a one-to-one correspondence with bistochastic $n \times n$ matrices [14]. Several properties and constructions for discrete copulas can be thus introduced by means of properties, notions and constructions of bistochastic matrices. For example, the product od copulas $C_1 * C_2$ introduced in [3] has its discrete counterpart $D_1 \star D_2$ described by the product of the corresponding bistochastic matrices. The class $\mathcal{D}_n$ of all discrete $I_n^2 \to [0,1]$ copulas is a polyhedron with vertices corresponding to the permutation matrices of order n. However, then the corresponding irreducible discrete copulas are just discrete copulas with range

I_n, as introduced in [16]. Each such copula related to a permutation σ describes the ordered statistics of x and y samples which are coupled together, i.e., if x_i is the j'th order statistics in the x sample, then y_i is the $\sigma(j)$'th order statistics in the y sample. For more details we recommend [19]. Similarly, m–dimensional case for $m > 2$ can be treated. Indeed, $D : I_n^m \to I_n$ is a discrete copula if and only if there are permutations $\sigma_1, \sigma_2, \ldots, \sigma_m$ of $(1, 2, \ldots, n)$ such that the sample

$$(x_{11}, \ldots, x_{1m}), \ldots, (x_{n1}, \ldots, x_{nm}),$$

with distinct values on each fixed coordinate, can be written in the form

$$(x'_{1\sigma_1(1)}, \ldots, x'_{1\sigma_m(1)}), \ldots, (x'_{n\sigma_1(n)}, \ldots, x'_{n\sigma_n(n)}),$$

where x'_{ij} is the jth order statistics in the sample from ith coordinate.

3 Extensions to copulas

Partial knowledge about the relationship of random variables X and Y restricts the choice of a copula C coupling X and Y. In several cases, such knowledge determines the values of C on a subset of domain $[0, 1]^2$ only, and we want to extend this information to an entire determining of C. Rarely such extension is unique, and thus we mostly look for some extremal (or simple) extensions. A typical case is when knowing the diagonal section $\delta : [0, 1] \to [0, 1]$ of a copula C, i.e., for $[0, 1]$ uniformly distributed random variables X and Y, knowing the distribution function of $Z = \max(X, Y)$. There always exists a copula whose diagonal section coincides with given δ, so–called diagonal copula [6, 22], given by

$$C_\delta(x, y) = \min\left(x, y, \frac{\delta(x) + \delta(y)}{2}\right).$$

Moreover, [2, 7, 10], there is always a weakest copula $B_\delta : [0, 1]^2 \to [0, 1]$ with $B_\delta(x, x) = \delta(x)$, $x \in [0, 1]$. The copula B_δ is given by

$$B_\delta(x, y) = \min(x, y) - \min_{t \in [x \wedge y, x \vee y]}(t - \delta(t)),$$

and called the Bertino copula.

In general, the strongest copula copula with given diagonal section δ need not exist. Observe that C_δ is always a maximal element of the class of copulas with diagonal section δ, and it is the strongest symmetric copula of that class.

The problem when a function $MT_\delta : [0, 1]^2 \to [0, 1]$,

$$MT_\delta(x, y) = \max(0, \delta(x \vee y) - |x - y|)$$

is a copula was solved in [4]. The function MT_δ is a copula (so–called Mayor–Torrens copula) if and only if the function $\delta - id$ is non–decreasing on $\delta^{-1}(]0, 1])$. Note that then $MT_\delta = B_\delta$ and

$$MT_\delta(x, y) = \min(x, y) - \min(x - \delta(x), y - \delta(y)).$$

Similar results were studied in the case of given opposite diagonal section $\omega : [0, 1] \to [0, 1]$, $\omega(x) = C(x, 1 - x)$, see [9, 10]. The function

$$C_\omega(x, y) = \max\left(0, x + y - 1, \frac{\omega(x) + \omega(y)}{2}\right)$$

is always a copula with opposite diagonal section ω. There is always a strongest copula with given ω, [9, 10]. Observe that the above results were straightforwardly shown in [9, 10], however, they can be derived from results for diagonal sections exploiting the constructions (3) or (4), see [11].

Recent results concerning the extensions from affine sections of copulas can be found in [13]. Moreover, in [12] we prepare extensions of horizontal sections of copulas, $h : [0,1] \to [0,1]$, $h(x) = C(x,b)$ for a fixed $b \in]0,1[$. For example, the function $C_h : [0,1]^2 \to [0,1]$ given by

$$C_h(x,y) = \begin{cases} \frac{h(x)y}{b} & \text{if } y \leq b \\ \frac{h(x)(1-y)+x(y-b)}{1-b} & \text{else,} \end{cases}$$

is always a copula with prescribed horizontal section h.
Note that related results for irreducible discrete copulas were discussed in [15, 21].

4 Compatibility of copulas

Any 3–copula $C : [0,1]^3 \to [0,1]$ induces three marginal 2–copulas C_{12}, C_{23}, $C_{13} : [0,1]^2 \to [0,1]$, i.e., from the dependence structure of a 3–variate random vector $V = (X,Y,Z)$ we can derive dependence structure of bivariate marginal random vectors (X,Z), (Y,Z) and (X,Z). Observe that not all triples of 2–copulas (E,F,G) can be joined by a common 3–copula C such that $E = C_{12}$, $F = C_{23}$ and $G = C_{13}$. In the positive case the triple of copulas (E,F,G) is called compatible. As a negative example recall the triple (W,W,W) while a triple (Π,Π,G) is compatible for any 2–copula G. Indeed, it is enough to put $C(x,y,z) = y\,G(x,z)$. Among few results in this domain recall the next results from [20, 24]:

(i) (M,F,G) is compatible if and only if $F = G$ and there is a unique 3–copula C,

$$C(x,y,z) = \max(0, M(x,y) + F(x,z) - x, M(x,y) + C(y,z) - y, C(x,z) + C(y,z) - z)$$

with margins (M,F,F).

(ii) (W,F,G) is compatible if and only if $G = F_-$, see (4), and the unique 3–copula C with margins (W,F,F_-) is given by

$$C(x,y,z) = \max(0, F(y,z) + x - 1, y + z - 1 - F(1-x,z)).$$

(iii) For any 2–copulas E,F, the triple of copulas $(E,F,E*F)$ is compatible (for the definition of $E*F$ see darsow) and as an example of an appropriate 3–copula C we can put

$$C(x,y,z) = \int_0^y \frac{\partial E(x,t)}{\partial t} \frac{\partial F(t,z)}{\partial t}\, dt.$$

(iv) If E or F is a shuffle of M then $G = E*F$ is the only 2–copula such that the triple (E,F,G) is compatible. Note that $E : [0,1]^2 \to [0,1]$ is a shuffle of M if and only if there is a (Lebesgue) measure preserving transformation $f : [0,1] \to [0,1]$ which is continuous on $[0,1]$ up to finitely many points, such that random variables X and Y uniformly distributed over $[0,1]$ and coupled by the copula E, are related by f, $Y = f(X)$.

5 Some open problems

In each of the above mentioned copula areas there are several open problems. We list some of them:

(5.1) Characterize discrete copulas $D : I_n \times I_m \to [0,1]$ for $n \neq m$.

(5.2) Is it possible to generalize the product $*$ of 2–copulas for n–copulas (product $\star$ of discrete 2–copulas to discrete n–copulas) ?

(5.3) For given b–horizontal and c–vertical sections of a 2–copula C, characterize all copulas with the same b–horizontal and c–vertical sections. Characterize functions h and v for which there is a copula C such that h is its horizontal section and v vertical section.

(5.4) Solve problem (5.3) for diagonal and opposite diagonal sections.

(5.5) For given 2–copulas E and F characterize all 2–copulas G such that the triple (E, F, G) is compatible.

(5.6) Characterize couples of Archimedean copulas (E, F) such that the triple (E, F, F) is compatible. For partial solutions see [5].

Acknowledgement

The support of the grant VEGA 1/2032/05 is kindly announced. This work was also supported by Science and Technology Assistance Agency under the contract No. APVT–20–003204.

References

[1] C. Alsina, R.B. Nelsen and B. Schweizer, On the characterization of a class of binary operations on distributions functions. *Stat. Probab. Lett.* **17** (1993) 85–89.

[2] S. Bertino, On dissimilarity between cyclic permutations. *Metron* **35** (1977) 53-88, in Italien.

[3] W.F. Darsow, B. Nguyen and E.T. Olsen, Copulas and Markov processes. *Illinois J. Math.* **36** (1992) 600-642.

[4] F. Durante, R. Mesiar, C. Sempi, On a family of copulas constructed from the diagonal section. *Soft Computing* **9** (2005), in press.

[5] P. Embrechts, F. Lindskog, A.J. McNeil, Modelling dependence with copulas and applications to risk management. In: *Handbook of Heavy Tailed Distributions in Finance*, S.T. Rachev, ed. Elsevier, Amsterdam, 2003, pp. 329-384.

[6] G.A. Fredricks, R.B. Nelsen, Copulas constructed from diagonal sections. In: *Distributions with Given Marginals and Moment Problems*, V. Beneš and J. Štepán, eds. Kluwer Academic Publishers, Dordrecht, 1997, pp. 129-136.

[7] G.A. Fredricks, R.B. Nelsen, The Bertino family of copulas. In: *Distributions with Given Marginals and Statistical Modelling*, C.M. Caudras et al., eds. Kluwer Academic Publishers, Dordrecht, 2002, pp. 81-91.

[8] E.P. Klement, R. Mesiar, E. Pap, Invariant copulas. *Kybernetika* **38** (2002) 275–285.

[9] E.P. Klement, A. Kolesárová, 1–Lipschitz aggregation operators, quasi–copulas and copulas with given diagonals. In: *Soft Methodology and Random Information Systems*, M. Lopéz–Díaz et al., eds., Springer–Verlag Berlin Heidelberg, 2004, pp. 205–211.

[10] E.P. Klement, A. Kolesárová, Extensions to copulas and quasi–copulas as special 1–Lipschitz aggregation operators. *Kybernetika* **41** (2005) 329-348.

[11] E.P. Klement, A. Kolesárová, Affine sections of 1–Lipschitz aggregation operators, quasi–copulas and copulas. Manuscript in preparation.

[12] E.P. Klement, A. Kolesárová, R. Mesiar, C. Sempi, Horizontal sections of copulas. Manuscript in preparation.

[13] A. Kolesárová, E.P. Klement, On affine sections of 1–Lipschitz aggregation operators. *Proc. EUSFLAT-LFA'2005*, Barcelona, September, 2005, CD.

[14] A. Kolesárová, R. Mesiar, J. Mordelová, C. Sempi, Discrete copulas, *IEEE Trans. on Fuzzy Systems*, accepted.

[15] A. Kolesárová, J. Mordelová, Quasi–copulas and copulas on a discrete scale. *Soft Computing* **9** (2005), in press.

[16] G. Mayor, J. Suñer, J. Torrens, Copula–like operations on finite settings. *IEEE Trans. on Fuzzy Systems*, accepted.

[17] G. Mayor, J. Torrens, Triangular norms on discrete settings. In: *Logical, Algebraic, Analytic, and Probabilistic Aspects of Triangular Norms*, E.P. Klement and R. Mesiar, eds., Elsevier, pp. 189–230, 2005.

[18] R. Mesiar, J. Szolgay, W-ordinal sums of copulas and quasi–copulas. *Proc. MAGIA-UWPM'2005*, Publishing House STU, Bratislava, 2005, pp. 78-83.

[19] R. Mesiar, Discrete copulas - What they are. *Proc. EUSFLAT-LFA'2005*, Barcelona, 2005, pp. 927–930.

[20] R. Mesiar, C. Sempi, A. Kolesárová, Compatibility of copulas. Manuscript in preparation.

[21] J. Mordelová, Some distinguished classes of 1–Lipschitz aggregation operators. *PhD thesis*, STU Bratislava, 2005.

[22] R.B. Nelsen, *An Introduction to Copulas*. Lecture Notes in Statistics 139, Springer Verlag, New York 1999.

[23] R.B. Nelsen, G.A. Fredricks, Diagonal copulas. In: *Distributions with given Marginals and Moment Problems*, V. Beneš and J. Štepán, eds. Kluwer Academic Publishers, Dordrecht, 1997, pp. 121-127.

[24] J.A Rodríguez-Lallena, M. Úbeda-Flores, Compatibility of three bivariate quasi–copulas: applications to copulas. In: *Soft Methodology and Random Information Systems*, M. Lopéz–Díaz et al., eds., Springer–Verlag Berlin Heidelberg, 2004, pp. 173-180.

[25] B. Schweizer, A. Sklar, *Probabilistic Metric Spaces*. North–Holland, New York, 1983.

[26] A. Sklar, Fonctions de répartition à n dimensions et leurs marges. *Publ. Inst. Statist. Univ. Paris* **8** (1959) 229–231.

A Modified Version of Vector Quantization for Incremental Clustering

Edwin Lughofer
{edwin.lughofer}@jku.at
Johannes Kepler University Linz
Altenbergerstrasse 69, A-4040 Linz, Austria

***Abstract*— Vector quantization is a popular and widely applied method for clustering data containing several data clouds, which should be well separated. However, it possesses some drawbacks especially in the case of online learning, which we want to omit in this paper. In this sense, we extend the conventional vector quantization by combining it with the idea of adaptive resonance theory network (ART) to form an incremental variant, which can be applied for online classification and approximation tasks with an unknown number of clusters. The second extension concerns the incorporation of the range of influence of clusters in the vector quantization learning process by selecting the 'winner neuron' based on the distances of a data point to the surface of all clusters. This neglects the generation of additional cluster centers within wide data clouds. The third extension introduce a deletion of so-called cluster satellites, i.e. very small clusters consisting of only a couple of data points and lying near a much bigger (= more significant) cluster. This can occur in the incremental version of vector quantization quite often, when centers move strongly over time due to newly loaded data. The problems of vector quantization and their improvements due to the modified version proposed in this paper are visualized based on two-dimensional data set examples as well as well-known higher dimensional clustering data sets. The impact of the modified variant onto the training of high-dimensional data-driven fuzzy systems, where clusters are back-projected to the axes in order to form fuzzy sets and rules, concludes the paper.**

***Index Terms*— vector quantization, clustering, adaptive resonance theory network, incremental, removing cluster satellites**

I. Introduction

Nowadays clustering plays an important role, whenever data bases or data sets should be divided into local areas and new observations be classified based on these local areas (denoting different states or patterns within the system). Other application tasks for clustering include the rule extraction for fuzzy models or forming neurons for neural networks. Data compression for huge data bases can be carried out as well. This is possible due to the fact that a cluster always can be seen as a group of data that are more similar to each other than data belonging to other clusters, see [1]. In this sense, a cluster center represents also a compact information about a local area or distinct data cloud within the data. A popular and widely applied clustering algorithm is the so-called vector quantization [2], which moves cluster centers denoted as neuron weights towards accumulation points in the data set, a more detailed description follows in Section II. In literature there are several extensions of vector quantization proposed, the most famous ones are the self organizing maps (SOM) going back to Kohonen [3] and neural gas network [4]. With the usage of both, different topologies can be exploited (such as circular or grid ones), where neurons which are neighbors in the network topology should possess similar weight vectors, i.e. cluster centers.

In this paper the original version of vector quantization is extended in three points. First, an incremental variant is demonstrated, which builds up clusters step by step and hence omits the pre-definition of the number of clusters, which has to be sent as parameter into conventional vector quantization. Second, a different distance strategy is incorporated by taking the distance of new incoming points to the range of influence of clusters and not to the cluster centers. In this sense, the range of influence in each direction is also calculated incrementally. Third, a satellite deletion strategy is proposed in order to remove not significant clusters (so-called satellites) after the complete learning process. This can be applied in connection with arbitrary crisp clustering techniques which generate cluster centers. All these three issues will be demonstrated in Section III. In Section IV the new modified variant of vector quantization will be compared with conventional vector quantization and some other well-known clustering methods based on well-known high-dimensional clustering data sets with respect to the quality of the obtained clusters measured by a well-known cluster validation index.

II. Vector Quantization

The purpose of vector quantization [2] originally stems from encoding discrete data vectors in order to compress data which has to be transferred quickly e.g. for online communication channels. The prototypes, here called neuron vectors, are the representatives for similar/nearby lying data vectors. From the mathematical point of view, vector quantization is basically a simplified version of k-means clustering in sample-mode adaptation, i.e. it is capable to update the parameters with each incoming data point. This should be not confused with a form of incremental clustering, where no iterations over the complete data set are possible. Let the dimensionality of the data to be clustered be p. The amount of neurons (clusters) C has to be parameterized a-priori, where each neuron has p parameters corresponding to the p components of each cluster center. With these notations and assuming that the input data is a-priori normalized due to its range, the algorithm for vector quantization can be formulated in the following way:

Algorithm 1: **Vector Quantization**

1) Choose initial values for the C neuron vectors $\vec{c}_i, i = 1, ..., C$, e.g. simply by taking the first C data points as cluster centers.
2) Fetch out the next data sample of the data set.
3) Calculate the distance of the selected data point to all cluster centers by using a predefined distance measure. Commonly, Euclidean distance is used.
4) Elicit the cluster center which is closest to the data point by taking the minimum over all calculated distances $\rightarrow$ winner neuron
5) Update the p components of the winner neuron by moving it towards the selected point $\vec{x}$:
$$\vec{c}_{win}^{(new)} = \vec{c}_{win}^{(old)} + \eta(\vec{x} - \vec{c}_{win}^{(old)}) \quad (1)$$
where the step size η has to be chosen a-priori and should be chosen appropriately. For example a value for η equal to 1 would move the winner neuron exactly to the selected data point in each iteration, a value near 0 would not give a significant change for the cluster center over all iterations.
6) If the data set contains data points which were not processed through steps 2 to 5, goto step 2, otherwise goto step 8
7) If any cluster center was moved significantly in the last iteration, say more than ϵ, reset the pointer to the data buffer at the beginning and goto step 2, otherwise stop

It has to be remarked that indeed mostly the Euclidean norm is used resulting in ellipsoidal clusters parallel to the main axes, but others are also possible: for instance with Mahalanobis distance ellipsoidal clusters in general position can be achieved [5] [6]. Furthermore, for a good convergence an adaptive learning gain η is recommended [7], which decreases with the number of iterations.

III. Modifications of Vector Quantization

In this sections necessary and recommended extensions are demonstrated in order to be able to apply vector quantization for online clustering tasks and to data sets with an unknown number of clusters. Moreover, an alternative distance strategy for omitting more cluster centers in wide data clouds and a cluster satellite deletion technique is demonstrated. The latter one assures that cluster artefacts which can come up during the incremental clustering process due to strong movements of cluster centers are deleted.

A. Vector Quantization in Incremental Mode

When inspecting Algorithm 1 it can be easily recognized that it is only applicable for offline clustering tasks as it processes through the entire data set more than one time. In this sense, it is not capable being applied for online processes, where incremental clustering is demanded, meaning clustering techniques which update their parameter for each newly loaded data block or even for each single data sample without taking account prior data. This is because Algorithm 1 iterates over the loaded data buffer several times. If this would be carried out for each incremental learning step i.e. for each actual loaded data block separately, the cluster centers would only represent a reliable partition of this data block and forget the older data completely. Indeed, it is theoretically possible to collect all the data points recorded or loaded so far, keep it in the virtual memory and perform a re-estimation of the cluster centers from time to time. This procedure, however, would result in an inacceptable computation performance, which could be verified in [8], when training fuzzy models from data with the help of vector quantization for input/output space partitioning.

Moreover, the number of clusters has to be known in advance, which can be a significant drawback, if the number of clusters is a-priori not known. Furthermore, in the case of online clustering the number of clusters are never known in advance as usually the data has to be sent into the algorithm as it is loaded (e.g. for online streams in data bases) or recorded (e.g. for online measurement systems). It should be noticed, that this problem can be indeed solved for the offline case by exploiting cluster validation indices [9] and applying them for different partitions obtained with different numbers of clusters. But, for the online case the drawback still remains. It could be also inspected that the convergence of vector quantization may be weak, when taking for instance the first C data points as the C cluster centers to start, see Section IV-A.

Hence, for omitting these drawbacks the idea of adaptive resonance theory network [10] is exploited. These networks consist of neurons which are able to adapt to new information without forgetting or overwriting already learned relationships, so to overcome the famous *stability/plasticity dilemma*. In ART networks, especially in the ART-2 algorithm, this conflict is solved by the introduction of a so-called *vigilance* parameter, which controls the tradeoff between adaptation of already learned clusters and generation of new clusters. In this sense, for each new data point the following condition is checked:

$$\|\vec{x} - \vec{c}_{win}\|_A \geq \rho \text{ and } \vec{x} \text{ is not faulty} \quad (2)$$

with $\vec{x}$ the actual data point, $\vec{c}_{win}$ the winning neuron and A the norm-inducing distance. If this condition is fulfilled, the prototype c_{C+1} of the new (the $C+1$th) cluster is set to the actual data point. In fact, it is true that the problem of a-priori defining the number of clusters C is shifted to finding a good value for the *vigilance* parameter ρ. But, a good guess for this parameter can be achieved much more easier, when clustering is applied onto data normalized into the hypercube $[0, 1]^p$: being far away from a new data point always can be explained with a certain distance value. In a trial and error tuning phase with quite a lot different data sets it turned out, that the following choice of this parameter should be preferred:

$$\rho = 0.3\frac{\sqrt{p}}{\sqrt{2}} \quad (3)$$

The dependency of ρ on the p-dimensional space diagonal, i.e. $\sqrt{p}$, can be explained with the so-called *curse of dimensionality* effect: the higher the dimension, the greater the distance between two adjacent data points, see [11]; therefore, the larger the parameter rho should get in order to prevent the algorithm to generate too much clusters and causing strong overfitting effects. The second part of condition (2) assures that

the new data point does not represent a faulty situation during data recordings. Indeed, it is very hard to decide whether a new incoming data point lying far outside previously loaded data clouds denotes a new operating condition, which should be in fact included into the clustering process, or a faulty situation, which should be not included. It is worthy to mention that in a fault detection framework [12] (where the modified version of vector quantization was applied for training fuzzy models) faulty points could be detected and marked before and hence omitted in the clustering process.

With these notations and by assuming normalized data vector quantization in incremental mode becomes:

Algorithm 2: **Vector Quantization in Incremental Mode**

1) Initialize the number of clusters to 0
2) Take the next incoming data point (online case) or fetch out a data sample from a data matrix randomly or ordered (offline case), let's call it $\vec{x}$
3) **If** the number of clusters is 0
   a) Set $i = 1$
   b) Set the first center c_1 to the actual data point, hence $\vec{c}_1 = \vec{x}$
   c) goto step 8
4) Calculate the distance of the selected data point to all cluster centers by using a predefined distance measure. Commonly, Euclidean or Mahalanobis distance is used.
5) Elicit the cluster center which is closest to the data point by taking the minimum over all calculated distances $\rightarrow$ winner neuron c_{win}
6) **If** $\|\vec{x} - c_{win}\|_A \geq \rho$ and $\vec{x}$ is not faulty
   a) Set $i = i + 1$
   b) Set $\vec{c}_i = \vec{x}$
   c) goto step 8
7) Update the $p + 1$ components of the winner neuron by moving it towards the selected point $\vec{x}$, as in (1)
8) If the data matrix still contains uncovered data (offline case) or new incoming data points are still available (online case) goto step 2, otherwise stop

From this definition it can be realized that each (newly loaded) data point is processed only once through the update process. Therefore, the learning gain η is not decreased by the amount of iterations as kept constant. This would result in a bad convergence of the algorithm, as no matter how many data points belong to one cluster (i.e. for which this cluster was the winning neuron) the shift of the center would be always to the same extent. A possibility for preventing this situation can be accomplished by steering η with the amount of data points belonging to each cluster in a monotonic decreasing way:

$$\eta_i = \frac{0.5}{k_i} \forall i \tag{4}$$

with k_i the number of data points belonging to cluster i. This is implemented in this way in Algorithm 2 and a reasonable choice as in k-means clustering algorithm [13] the step size is also normalized by this number, whereas original vector quantization is a simplified version of k-means clustering.

B. An Alternative Distance Strategy

The problem with vector quantization in incremental mode is that in the case of wider data clouds or points belonging together widely spread over the input space Algorithm 2 tends to generate more clusters than necessary and hence perform an 'overclustering' and incorrect partition of the input space. This fact is underlined in the left image of Figure 1, where obviously three clusters are the optimal case, but five cluster are generated. This is because for the big data pattern three clusters are produced instead of one.

The reason for this unpleasant occurrence lies at hand: Algorithm 2 (as well as conventional vector quantization, of course) compares each new incoming point with all the cluster centers, which can happen to be far away even if the data point is close to its spanned range of influence represented by those data points already belonging to the cluster. Therefore, an obvious overcoming of this drawback can be achieved by calculating the range of influence during the incremental learning process and taking the distance of new points to these ranges instead of to the cluster centers. Whenever the Euclidean distance is used as distance measure (which is more or less the most common choice), axis-parallel ellipsoids are triggered as clusters, whose range of influence (as 2σ-area) can be calculated in incremental mode by exploiting recursive variance formula [14]:

$$\begin{aligned} &k_{win}\sigma^2_{win,j}(new) = \\ &(k_{win} - 1)\sigma^2_{win,j}(old) + k_i \Delta c^2_{win,j} + (c_{win,j} - x_j)^2 \quad \forall j \end{aligned} \tag{5}$$

where $\Delta c_{win,j}$ is the distance of the old prototype to the new prototype of the cluster nearest to the actual point $\vec{x}$ in the jth dimension and k_{win} is the amount of data points lying nearest to cluster c_{win} and can therefore be simply updated through counting. For the distance of the new data point to the surface of the multi-dimensional ellipsoid spanned by a cluster we take the distance along the direction from the actual point towards the cluster center.

Lemma 1: Let therefore be $\sum_{j=1}^{p} \frac{(x_j - c_{ij})^2}{\sigma^2_{ij}} = 1$ be a multidimensional ellipsoid of the ith cluster in main position, i.e. the axis of the ellipsoid parallel to the axis of the coordinates, σ_{ij} the variance of the data belonging to the ith cluster (i.e. for which the ith cluster was the winning neuron) in dimension j, then the Euclidian distance of the new data point $(q_1, ..., q_p)$ to the surface along the direction towards the cluster center of the cluster center c_{ij} is given by

$$dist = (1 - t)\sqrt{\sum_{j=1}^{p}(q_j - c_{ij})^2} \tag{6}$$

with

$$t = \frac{1}{\sqrt{\sum_{j=1}^{p} \frac{(q_j - c_{ij})^2}{\sigma^2_{ij}}}} \tag{7}$$

Proof: Let $\vec{x} = \vec{c_i} + t(\vec{q} - \vec{c_i})$ the straight line in the multi-dimensional space between new point $(q_1, ..., q_p)$ and the ith cluster center, then the crossing point of this straight line with the multi-dimensional ellipsoid is obtained

by setting the parameter vector into the ellipsoid equation, hence $\sum_{j=1}^{p} \frac{t^2(q_j - c_{ij})^2}{\sigma_{ij}^2} = 1$, and by solving this equation after t, achieving (7). The Euclidian distance between the crossing point and $(q_1, ..., q_p)$ is computed by summing up the squared differences $q_j - c_{ij} - t(q_j - c_{ij}) = q_j(1-t) - c_{ij}(1-t)$ for all j, resulting in (6). ■

In fact, the distance (6) is only computed for the actual data point, if it is lying outside the ranges of influence of all clusters, i.e. the condition

$$\exists i \quad \sum_{j=1}^{p} \frac{(q_j - c_{ij})^2}{\sigma_{ij}^2} \leq 1 \tag{8}$$

is not fulfilled. Otherwise, the usual distance strategy is applied for all clusters, whose range of influence the actual data point lies inside.

This leads us to the modified version of vector quantization in incremental mode:

Algorithm 3: **Modified Version of Vector Quantization in Incremental Mode**

1) Initialize the number of clusters to 0
2) Take the next incoming data point (online case) or fetch out a data sample from a data matrix randomly or ordered (offline case), let's call it $\vec{x}$
3) **If** the number of clusters is 0
   a) Set $i = 1$
   b) Set the first center c_1 to the actual data point, hence $\vec{c}_1 = \vec{x}$, set $\vec{\sigma}_1 = \vec{0}$
   c) goto step 8
4) **If** the actual data point lies inside any cluster's range of influence, i.e. the condition (8) is fulfilled for at least one i (where σ_{ij} are artificially taken as $\max(\sigma_{ij}, \epsilon)$ with $\epsilon > 0$ in order to stay numerically stable)
   a) Calculate the distance of the selected data point to all those cluster centers fulfilling (8) by using Euclidian distance measure.
   b) Elicit the cluster center which is closest to the data point by taking the minimum over all calculated distances → winner neuron c_{win}
   c) Set $min_{dist} = 0$
5) **Else If** the actual data point lies outside of all cluster's range of influence, i.e. condition (8) is not fulfilled for any cluster
   a) Calculate (6) for all clusters
   b) Elicit the cluster center which is closest to the data point by taking the minimum over all calculated distances → winner neuron c_{win}
   c) Set min_{dist} as the minimum over all distances
6) **If** $min_{dist} \geq \rho$ and $\vec{x}$ is not faulty
   a) Set $i = i + 1$
   b) $\vec{c}_i = \vec{x}$, $\vec{\sigma}_i = \vec{0}$
   c) goto step 8
7) Update the $p + 1$ components of the winner neuron by moving it towards the selected point $\vec{x}$ as in (1), update the variance in each direction by using (5).
8) If the data matrix still contains uncovered data (offline case) or new incoming data points are still available (online case) goto step 2, otherwise stop

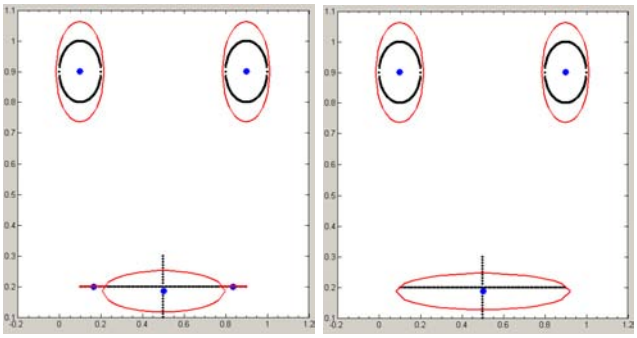

Fig. 1. Left Image: Clustering obtained by Algorithm 2 → too many clusters Right image: Clustering obtained by its modified version with new distance strategy (Algorithm 3) → clusters are ok

Figure 1 demonstrates the impact of this modified version of vector quantization. While Algorithm 2 generates new clusters for data points lying near the range of influence of another cluster (left image), the modified version performs better and extends the range of influence (drawn as the 2σ-range in each dimension) of the nearby lying cluster (right image). The cluster centers are visualized as big dark data dots. In fact, in this simple example the generation of too much clusters could be also prevented for Algorithm 2 by choosing a higher value for ρ, but this cannot not be reasonably accomplished for online data streams in advance or also for arbitrary high-dimensional data sets with an unknown number of clusters: the wideness of such big data clouds can be even bigger and too high values of ρ would usually trigger too less clusters.

It should be noticed that Algorithm 3 can be extended straightforward to the case when Mahalanobis distance measure is used, which triggers ellipsoidal clusters in general position. For this, the recursive covariance formula, which is similar to the recursive variance formula, can be exploited for updating the covariance matrix of the data. Furthermore, Lemma 1 needs to be extended with an appropriate distance calculation from a new data point to the surface for an ellipsoidal in general position.

C. Satellite Deletion

In the case of online learning the incremental training scheme in Algorithm 3 may lead to undesirable tiny clusters, so-called cluster satellites, which lie very close to significantly bigger ones. An example is demonstrated in the left image in Figure 2 (the data taken from MATLAB's cluster demo data), where the tiny cluster is obviously superfluous. The reason for this unpleasant effect is the following (could be also observed in other examples): at the beginning of the incremental training process the first points forming the bottom cluster appear at the upper region of this cluster, the cluster had a very narrow range of influence at this stage (see small ellipsis inside and surrounding a bigger dark dot). Afterwards a data point at the lower region came in and formed a new cluster, as being too far away from the small ellipsis in the upper region. This cluster forming at this stage of the incremental (online) learning process was correct, as in online mode the data comes as it comes and a foreseeing into the future is not possible.

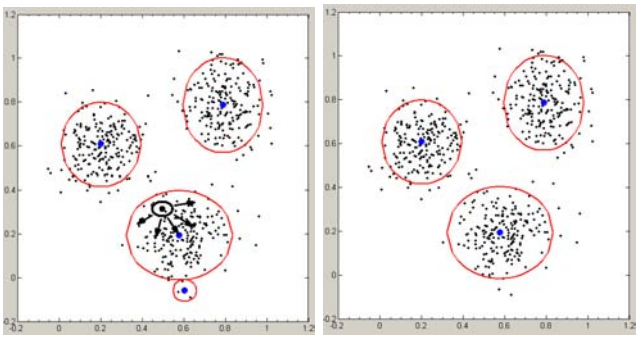

Fig. 2. Left Image: An unpleasant satellite cluster by applying Algorithm 3; right image: The unpleasant satellite cluster removed by applying Algorithm4

However, 'overclustering' should be prevented as this leads to false information when for instance performing classifications based on the clustering. In this sense, the following strategy was developed for deleting so-called cluster satellites, which can be applied after Algorithm 3, so at the end of the whole learning process, as no prior loaded data is needed.

Algorithm 4: **Satellite Deletion**

1) **For** all $i = 1, ..., C$ clusters generated by Algorithm 3 perform the following steps:
2) **If** $\frac{k_i}{N} < 1\%$ where k_i the amount of data belonging to the ith cluster and N the number of data points loaded in sum
   a) Mark the ith cluster for cutting out, as it only captures outliers
   b) Continue with next cluster
3) **If** $\frac{k_i}{N} < low\_mass$
   a) **If** the ith cluster center lies inside the range of influence of any other cluster, let's say cluster k
      i) If $\frac{\sum_{j=1}^{p} \sigma_{ij}}{\sum_{j=1}^{p} \sigma_{kj}} < \epsilon$, where σ_{ij} the jth axes of the ellipsoid spanned by the ith cluster, then mark the ith cluster for cutting out
   b) **Else**
      i) Calculate the distance of the ith cluster center to the surface of all other clusters after (6) in Lemma 1
      ii) Elicit the cluster which is closest to the cluster center of the ith cluster, let's say cluster k
      iii) If $\frac{\sum_{j=1}^{p} \sigma_{ij}}{\sum_{j=1}^{p} \sigma_{kj}} < \epsilon$, where σ_{ij} the jth axes of the ellipsoid spanned by the ith cluster, then mark the ith cluster for cutting out
4) **End For**
5) Cut out all marked clusters

This strategy is based on the investigations, what characterizes in fact a satellite cluster, namely 1.) a low mass i.e. a small fraction of data points belonging to it, 2.) the cluster center has to be close or inside the range of influence of the (most) adjacent cluster, 3.) the cluster has to be significantly smaller in its range of influence than the (most) adjacent cluster. Moreover, cluster with a very low mass will be cut out immediately (see Step 2), as they denote usually outliers. The marking and not directly cutting out the clusters ensures that satellite clusters of satellite clusters are cut out, too.

Note that in Algorithm 4 satellite clusters are completely cut out. This is opposed to the situation where two or more significant clusters move together and an intrinsic penetration of spheres or ellipsoids can be observed. In this case, which can quite often occur during incremental learning steps, well-known cluster merging algorithms can be applied, for instance [15], [16].

TABLE I

QUALITY OF CLUSTERS DENOTED BY THE PS-INDEX OBTAINED BY FIVE DIFFERENT CLUSTERING METHODS INCLUDING VQ-INC-MOD

Method	2-dim Data Set 1	2-dim Data Set 2	Iris Data	Wine Data
k-means	1.9379 / 3	2.098 / 3	1.38 / 3	1.57 / 3
subclust	1.934 / 2	2.024 / 3	1.10 / 3	1.75 / 3
VQ	1.9370 / 3	2.214 / 3	-0.16 / 3	1.31 / 3
VQ-INC	0.004 / 5	-0.5841 / 6	-0.03 / 5	-8.59 / 5
VQ-INC-MOD	1.935 / 3	2.167 / 3	1.14 / 3	1.77 / 3

IV. EVALUATION

In this section an evaluation of the new modified variant of vector quantization (Algorithm 3 denoted as VQ-INC-MOD) in connection with satellite deletion (Algorithm 4) is given. The conventional incremental version of vector quantization will be also tested and denoted as VQ-INC. This evaluation demonstrates the performance on some clustering benchmark data sets as well as the applicability for generating high-dimensional fuzzy models by cluster projection onto the axes. For the first case the new approach is compared with conventional vector quantization as well as some well-known clustering methods based on the cluster validation index by Yang and Wu, the so-called PS-index [17]. For the high-dimensional modelling tasks the obtained model complexities and accuracies are compared between the different clustering methods.

A. Results on Clustering Data Sets

In Table I a comparison between different clustering methods (columns) performing on various data sets (rows) is demonstrated, the entries ahead the slashes in the matrix correspond to the value obtained from the PS-Index: the higher this value, the better the quality of the clusters, i.e. the better the centers are set. If for example the correct number of clusters present (and known) in the data set is missed, this index will deliver a significant lower value than in the case of the optimal amount. The entries after the slashes are the number of clusters produced by the methods, whereas in all cases three clusters are the optimal choice.

For comparison reasons all the clustering methods in Table I are crisp clustering methods, which produce centers and hence are mainly applicable for finding clearly separable data clouds in a data set. It should be noticed that for k-means [13] and conventional vector quantization [2] the number of clusters have to be set in advance, which is not the case for subtractive clustering [18] and the modified versions of vector

quantization proposed in this paper. Both, Algorithm 2 (VQ-INC) as well as Algorithm 3 (VQ-INC-MOD) are connected with the satellite deletion strategy as defined in Algorithm 4. Note that the later can be applied at the end of any batch or incremental clustering process, which delivers clusters with centers and ranges of influence in each direction. The chosen data sets include two two-dimensional ones, the first one is the same as visualized in Figure 1, the second one is the obtained set when extracting first and third column in MATLAB's *clusterdemo.dat* file. The later possesses three clusters, where two are partially merged, but still clearly visible by human eyes. Obviously, for both data sets the new Algorithm 3 can compete with all the other clustering procedures. Subtractive clustering could only produce two clusters or more for the upper two circles.

For an evaluation based on high-dimensional data the two famous data sets *Iris* and *Wine* data (available in the UCI repository, see http://www.ics.uci.edu/ mlearn/MLRepository.html) are applied. While *Iris* is a four dimensional data set, the *Wine* data includes 13 dimensions; both data sets contain three classes and hence the optimal number of clusters is set to 3 for k-means and conventional vector quantization. For the *Iris* data set the performance of Algorithm 3 is second among all methods and slightly worse than the best one, namely k-means, whereas for the incremental version of vector quantization using the usual distance strategy (Algorithm 2) totally fails as producing an unacceptable number of clusters (5). No cluster had to be merged after the entire clustering process. For the *Wine* data the new modified version of vector quantization can even outperform all of the other methods with respect to the cluster quality. This is quite a strong result for the satellite deletion strategy as for this data set VQ-INC-MOD originally produced 13 different cluster, where 10 represented just outliers and no significant clusters. These could be removed by applying the satellite deletion strategy as proposed in Algorithm 4. Again Algorithm 2 could not find the correct number of clusters and produced an unacceptable result. It should be noticed, that for all data sets the standard parameter settings for η_i and ρ as presented in (4) and (3) are applied, so no parameter tuning steps took place.

In Figure 3 data from a practical application example is visualized: this data represents the dependency between immersion depth (x-axis) and the actual angle of a metal sheet at a steel plate bending machine (y-axis). This dependency is based on data from 10 different manufacturers for the same material which should be bent, the corresponding lines (as data point connections) visualized in different color styles. From this figure it is clearly visible that three data clouds can be separated, the first one data cloud represented by only the data from one manufacturer (dark dotted lines in the middle), the second one as lines above represented by data from two other manufacturers and the third one as lines below represented by data from the remaining seven manufacturers. We could realize, that, when adding three inputs for building a more accurate prediction model for the bending angle (as a two-dimensional linear model did not yield sufficient accuracy),

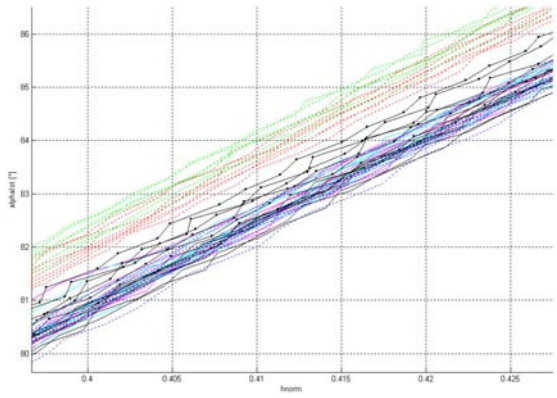

Fig. 3. Data from different manufacturers represented as different colored lines

this three areas represent a projection of three distinct data clouds onto the two-dimensional space. VQ-INC-MOD could generate exactly three clusters and therefore it was possible to obtain a well-interpretable accurate fuzzy model with three rules only, as the clusters were back-projected to the one-dimensional axes to form the fuzzy sets and the rules' premise parts. This was again possible with the standard parameter setting for the sensible distance parameter ρ (which steers the cluster adjoinment during incremental training). It could be also verified that when reducing the value of this parameter to the half of its default value, so in our case of a five-dimensional system after (3) to $0.3\frac{\sqrt{5}}{\sqrt{2}}/2 = 0.237$ still three clusters and hence rules could be obtained. This shows us a quite good stability of the new clustering method with respect to the choice of the parameter ρ.

B. Applying VQ-INC-MOD for the Generation of High-Dimensional Fuzzy Models

In order to demonstrate practical feasibility in an industrial process VQ-INC-MOD is applied for generating high-dimensional fuzzy models from measurement data. This is accomplished on the basis of cluster back-projection onto the one-dimensional space as it is also carried out in approaches such as FMCLUST [6] or genfis2 [18] implemented in MATLAB's fuzzy logic toolbox. We exchange subtractive clustering in genfis2 one time with VQ-INC, the other time with VQ-INC-MOD and compare the performance of these three methods onto high-dimensional measurement data from an engine test bench with respect to quality and complexity of the models. Real faults were simulated at the test bench while recording the data, which should be detected on the basis of the trained fuzzy models by calculating the deviation of actual measurements to the models. Hence, the detection rate, i.e. the number of correctly detected faults is stated as additional measure in Table II as well (note that the overdetection rate was always 0% for all approaches and is therefore not listed here). In sum, 56 up to five-dimensional reasonable models could be extracted automatically from the data with the help of variable selection methods [19], where 70 channel were measured in

TABLE II
COMPARISON OF CLUSTERING ALGORITHMS IN CONNECTION WITH FUZZY SYSTEM TRAINING, FIRST TABLE: OFFLINE CASE, SECOND TABLE: ONLINE CASE

Method	Complexity	Quality	Det. Rate
subclust	5.07	0.9201	70% / 75%
VQ-INC	4.5	0.941	72.5% / 75%
VQ-INC-MOD	3.25	0.942	77.5% / 87.5%

Method	Complexity	Quality	Det. Rate
subclust + consequ. adapt	9.43	0.793	53.75% / 50%
VQ-INC + FLEXFIS	8.24	0.934	66.25% / 75%
VQ-INC-MOD + FLEXFIS	7.12	0.936	70% / 75%

sum. This gives a good coverage of channels, when taking into account, that some of the remaining 14 appear in the input side of the models. The table is split into two parts: the first one represents the offline case, i.e. models are generated in advance in an offline step and new online measurements are checked; the second one demonstrates the online case, where fuzzy models have to be adaptively trained and extended. A high-frequented re-building of all the 56 models is not possible, as this would slow down the whole process significantly, such that real time performance cannot be achieved, see [8]. In the case of genfis2 an adaptation of linear rule consequent parameter alone has to be carried, as subtractive clustering is a batch clustering variant with no cluster (= rule) adjoining strategy. This is different for VQ-INC and VQ-INC-MOD, which build up the clusters (and therefore rules and fuzzy sets) in an incremental manner. A stable connection of premise part and antecedent part learning in incremental manner is carried out as described in [8], denoted as *FLEXFIS*. From this point of view, it is quite obvious, that genfis2 with subtractive clustering fails completely opposed to genfis2 with VQ-INC and VQ-INC-MOD, see quality and detection rates in the second part of Table II. Of course, all the online results are behind the offline results, as point per point is loaded into the training algorithm. For the offline case conventional genfis2 performs better, but is still behind the other two approaches with respect to both, quality and complexity. The complexity is measured by the average number of rules over the 56 fuzzy models. All the detection rates are measured on two bases: the first number corresponds to the measurement basis i.e. all by a fault affected measurements are counted (in sum 80), the second one corresponds to the fault bases, i.e. all different kind of faults are counted (in sum eight). It should be noticed that for VQ-INC and VQ-INC-MOD again the standard values for the parameters η_i and ρ were applied.

V. CONCLUSION

A new modified version of vector quantization (VQ-INC-MOD) was proposed. It extends conventional vector quantization to an incremental variant and incorporates a new distance strategy, which takes into account the range of influence of clusters. In this sense, the generation of more clusters within wide data clouds is prevented. A satellite deletion strategy can be appended to any clustering technique producing cluster centers and the range of influence of clusters. This is for removing not really significant clusters or clusters representing just outliers in the data set, which can come up during incremental learning as the future data points are unknown at each learning step. When inspecting the results in Sections IV-A and IV-B, it can be realized that VQ-INC-MOD can compete with conventional clustering methods with respect to the accuracy and quality of the clusters, (represented by a well-known cluster validation index) and fuzzy models obtained via cluster projection for forming the premise part and least squares antecedent learning afterwards. This is quite a strong result, as the new method acts on the data set on a point-per-point basis and therefore can be applied for fast online application tasks or huge data bases.

REFERENCES

[1] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. U.S.A.: Kluwer Academic/Plenum Publishers, ISBN 0-306-40671-3, July 1981.

[2] R. Gray, "Vector quantization," *IEEE ASSP Magazine*, pp. 4–29, 1984.

[3] T. Kohonen, *Self-Organizing Maps*. Berlin Heidelberg, Germany, second extended edition: Springer Verlag, 1995.

[4] T. Martinetz, S. Berkovich, and K. Schulten, "Neural gas network for vector quantization and its application to time-series prediction," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 558–569, 1993.

[5] D. Gustafson and W. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," in *Proc. IEEE CDC*, San Diego, CA, USA, 1979, pp. 761–766.

[6] R. Babuska and H. Verbruggen, "Constructing fuzzy models by product space clustering," in *Fuzzy Model Identification: Selected Approaches*, H. Hellendoorn and D. Driankov, Eds. Berlin: Springer, 1997, pp. 53–90.

[7] O. Nelles, *Nonlinear System Identification*. Germany: Springer Verlag Berlin, 2001.

[8] E. Lughofer and E. Klement, "FLEXFIS: A variant for incremental learning of Takagi-Sugeno fuzzy systems," in *Proceedings of FUZZ-IEEE 2005*, Reno, Nevada, U.S.A., 2005.

[9] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, no. 2/3, pp. 107–145, 2001.

[10] G. A. Carpenter and S. Grossberg, "Adaptive resonance theory (art)," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, MA: MIT Press, 1995, pp. 79–82.

[11] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. New York, Berlin, Heidelberg, Germany: Springer Verlag, 2001.

[12] E. Lughofer, E. Klement, J. Lujan, and C. Guardiola, "Model-based fault detection in multi-sensor measurement systems," in *Proceedings of IEEE IS 2004*, Varna, Bulgaria, 2004, pp. 184–189.

[13] K. Alsabti, S. Ranka, and V. Singh, "An efficient k-means clustering algorithm," in *IPPS: 11th International Parallel Processing Symposium*. IEEE Computer Society Press, 1998. [Online]. Available: citeseer.nj.nec.com/alsabti98efficient.html

[14] S. Qin, W. Li, and H. Yue, "Recursive PCA for adaptive process monitoring," *Journal of Process Control*, vol. 10, pp. 471–486, 2000.

[15] R. Krishnapuram and C. Freg, "Fitting an unknown number of lines and planes to image data through compatible cluster merging," *Pattern Recognition*, vol. 25, no. 4, pp. 385–400, 1992.

[16] F.-H. Rhee and B.-I. Choi, "A convex cluster merging algorithm using support vector machines," in *Proceedings of the 12th IEEE International Conference on Fuzzy Systems*, vol. 2, 2003, pp. 892–895.

[17] M. Yang and K. Wu, "A new validity index for clustering," in *Proceedings of the IEEE International Conference on Fuzzy Systems*, Melbourne, Australia, 2001.

[18] S. Chiu, "Fuzzy model identification based on cluster estimation," *Journal of Intelligent and Fuzzy Systems*, vol. 2, no. 3, 1994.

[19] W. Groißböck, E. Lughofer, and E. Klement, "A comparison of variable selection methods with the main focus on orthogonalization," in *Soft Methodology and Random Information Systems*, ser. Advances in Soft Computing, M. Lopéz-Díaz, M. Gil, P. Grzegorzewski, O. Hryniewicz, and J. Lawry, Eds. Berlin, Heidelberg, New York: Springer, 2004, pp. 479–486.

Actual results and open problems in the theory of generalized analysis

Endre Pap

Department of Mathematics and Informatics, University of Novi Sad
Trg Dositeja Obradovića 4, 21000 Novi Sad, Serbia and Montenegro,
e-mail: pape@eunet.yu

1 Introduction

Building models in practice often we are facing with nonlinearity and uncertainty and we are trying to find the optimal model. For that purpose usually we have to use different mathematical tools. We will present here some results on a generalization of the real mathematical analysis, the so called pseudo-analysis. For the range of functions and measures instead of the field of real numbers it is taken a semiring on the real interval $[a,b] \subset [-\infty,+\infty]$, denoting the corresponding operations as $\oplus$ (pseudo-addition) and $\odot$ (pseudo-multiplication). There are many applications of this theory modeling nonlinearity, uncertainty in many optimization problems, nonlinear partial differential equations, nonlinear difference equations, optimal control, fuzzy systems, see [3, 6, 9, 16, 17].

2 Basics of pseudo-analysis

Let $[a,b]$ be a closed (in some cases semiclosed) subinterval of $[-\infty,+\infty]$. We consider here a total order $\leq$ on $[a,b]$ (although it can be taken in the general case a partial order). The operation $\oplus$ (pseudo-addition) is a function $\oplus : [a,b] \times [a,b] \to [a,b]$ which is commutative, nondecreasing, associative and has a zero element, denoted by $\mathbb{0}$. Let $[a,b]_+ = \{x : x \in [a,b],\ \ x \geq \mathbb{0}\}$. The operation $\odot$ (*pseudo-multiplication*) is a function $\odot : [a,b] \times [a,b] \to [a,b]$ which is commutative, positively nondecreasing, i.e. $x \leq y$ implies $x \odot z \leq y \odot z,\ z \in [a,b]_+$, associative and for which there exist a unit element $\mathbb{1} \in [a,b]$, i.e., for each $x \in [a,b]$, $\mathbb{1} \odot x = x$. We suppose, further, $\mathbb{0} \odot x = \mathbb{0}$ and that $\odot$ is a distributive pseudo-multiplication with respect to $\oplus$, i.e., $x \odot (y \oplus z) = (x \odot y) \oplus (x \odot z)$. The structure $([a,b], \oplus, \odot)$ is a *semiring* (we can consider semirings in general settings on an arbitrary set endowed with two operations satisfying the previously mentioned properties). Here we will take as basic the following special real semirings (using the equality $\mathbb{0} \odot x = \mathbb{0}$ we can consider also closed intervals in the following examples): **Case I)** Pseudo-addition is idempotent and pseudo-multiplication is not idempotent, e.g., $x \oplus y = \min(x,y), \quad x \odot y = x + y$, on the interval $]-\infty,+\infty]$. We have $\mathbb{0} = +\infty$ and $\mathbb{1} = 0$.
Case II) Semirings with pseudo-operations defined by monotone and continuous generator g. In this case we will consider only strict pseudo-addition, i.e., such that the function $\oplus$

is continuous and strictly increasing in $]a,b[\times]a,b[$ and therefore there exists a monotone function g (generator for $\oplus$), $g:[a,b]\to[-\infty,\infty]$ (or with values in $[0,\infty]$) such $g(\mathbb{0})=0$ and $u\oplus v=g^{-1}(g(u)+g(v)),\quad u\odot v=g^{-1}(g(u)g(v))$.
Case III) Both pseudo-addition and pseudo-multiplications are idempotent, e.g., let $\oplus=\max$ and $\odot=\min$ on the interval $[-\infty,+\infty]$.

An idempotent semigroup (semiring, e.g., cases I and III) P is called an *idempotent metric semigroup (semiring)* if it is endowed with a metric $d:P\times P\to\mathbb{R}$ such that the operation $\oplus$ is (respectively, the operations $\oplus$ and $\odot$ are) uniformly continuous on any order-bounded set in the topology induced by d and any order-bounded set is bounded in the metric. Let X be a set, and let $P=(P,\oplus,d)$ be an idempotent metric semigroup. The set $B(X,P)$ of bounded mappings $X\to P$, i.e., mappings with order-bounded range, is an idempotent metric semigroup with respect to the pointwise addition $(\varphi\oplus\psi)(x)=\varphi(x)\oplus\psi(x)$, the corresponding partial order, and the uniform metric $d(\varphi,\psi)=\sup_x d(\varphi(x),\psi(x))$. If $P=(P,\oplus,\odot,\rho)$ is a semiring, then $B(X,P)$ has the structure of an *P-semimodule*, i.e., the multiplication by elements of P is defined on $B(X,P)$ by $(a\odot\varphi)(x)=a\odot\varphi(x)$. This P-semimodule will also be referred to as the space of (bounded) P-valued functions on X. If X is a topological space, then by $C(X,P)$ we denote the subsemimodule of continuous functions in $B(X,P)$. If X is finite, $X=\{x_1,\dots,x_n\}$, $n\in\mathbb{N}$, then the semimodules $C(X,P)$ and $B(X,P)$ coincide and can be identified with the semimodule $P^n=\{(a_1,\dots,a_n):a_j\in P\}$. Any vector $a\in P^n$ can be uniquely represented as a linear combination $a=\bigoplus_{j=1}^n a_j\odot e_j$, where $\{e_j,\ j=1,\dots,n\}$ is the standard basis of P^n (the jth coordinate of e_j is equal to $\mathbb{1}$, and the other coordinates are equal to $\mathbb{0}$). As in the classical linear algebra, we can readily prove that the semimodule of continuous homomorphisms $h:P^n\to P$ (in what follows such homomorphisms are called *pseudo linear functionals* on P^n) is isomorphic to P^n itself. Similarly, any endomorphism $H:P^n\to P^n$ (a linear operator on P^n) is determined by an P-valued $n\times n$ matrix, see [3, 9].

There are further generalizations related the pseudo-operations in the theory of pseudo-analysis. Related to further generalization of the real semiring structure to the case when the operations $\oplus$ and $\odot$ are noncommutative and non-associative, and the (left) right distributivity of $\odot$ over $\oplus$ plays a crucial rule. There is obtained a representation theorem for such operations and there is given a complete characterization for generalized pseudo-addition and pseudo-multiplication, see [22]. The second generalization is related to relaxation of the distributivity law, see [7], which enables a generalization of the classical von Neumann-Morgenstern utility theory to hybrid probability-possibilistic utility theory, see [4]. The third generalization consists in some symmetrization of maximum and minimum, used in an integral representation of the utility functional in [19].

Let X be a non-empty set. Let $\mathcal{A}$ be a σ-algebra of subsets of X. We shall suppose that $m:\mathcal{A}\to[a,b]_+$ be a σ-$\oplus$-additive measure. The construction of pseudo-integral, denoted by $\int_X^{\oplus} f\odot dm$, is similar to the construction of the Lebesgue integral, see [16, 17]. Then it can be introduced pseudo-convolution and pseudo-Laplace transform with may different applications, see [17, 20, 21].

3 A representation theorem in Idempotent Analysis

We present here the simplest versions of some general facts of idempotent analysis, in particular, restricting our consideration to the case of standard $(\min, +)$ semiring P. Proofs, generalizations and references could be found in [9, 10, 11, 12].

All idempotent measures are absolutely continuous; i.e., any such measure can be represented as the idempotent integral of a density function with respect to some standard measure. Let us formulate this fact more precisely. Let $C_{\mathbb{0}}(X, P)$ denote the space of continuous functions $f : X \to P$ on a locally compact normal space X vanishing at infinity (i.e. such that for any $\varepsilon > 0$ there exists a compact set $K \subset X$ such that $d(\mathbb{0}, f(x)) < \varepsilon$ for all $x \in X \setminus K$). The topology on $C_{\mathbb{0}}(X, P)$ is defined by the uniform metric $d(f, g) = \sup_X d(f(x), g(x))$. The space $C_{\mathbb{0}}(X, P)$ is an idempotent semimodule. If X is a compact set, then the semimodule $C_{\mathbb{0}}(X, P)$ coincides with the semimodule $C(X, P)$ of all continuous functions from X to P. The homomorphisms $C_{\mathbb{0}}(X, P) \to P$ will be called *pseudo linear functionals* on $C_{\mathbb{0}}(X, P)$. The set of linear functionals will be denoted by $C^*_{\mathbb{0}}(X, P)$ and called the *dual semimodule* of $C_{\mathbb{0}}(X, P)$.

Theorem 1 *For any $m \in C^*_{\mathbb{0}}(X, P)$ there exists a unique lower semicontinuous and bounded below function $f\colon X \to P$ such that*

$$m(h) = \inf_x f(x) \odot h(x) \qquad \forall h \in C_{\mathbb{0}}(X, P). \tag{1}$$

Conversely, any function $f\colon X \to P$ bounded below defines an element $m \in C^*_{\mathbb{0}}(X, P)$ by formula (1). At last, the functionals m_{f_1} and m_{f_2} coincide if and only if the functions f_1 and f_2 have the same lower semicontinuous closures; that is, $\operatorname{Cl} f_1 = \operatorname{Cl} f_2$, where

$$(\operatorname{Cl} f)(x) = \sup\{\psi(x) : \psi \le f,\, \psi \in C(X, P)\}.$$

The Riesz–Markov theorem in functional analysis establishes a one-to-one correspondence between continuous linear functionals on the space of continuous real functions on a locally compact space X vanishing at infinity and regular finite Borel measures on X. Similar correspondence exists in idempotent analysis. We can define an idempotent measure μ_f on the subsets of X by the formulas $\mu_f(A) = \inf\{x : x \in A\}$. This measure is σ-$\oplus$-additive. Equation (1) specifies a continuation of m_f to the set of P-valued functions bounded below. On analogy with conventional analysis, we say that such functions are integrable with respect to the measure μ_f and denote the values taken by m_f on these functions by the *idempotent integral*

$$m_f(h) = \int_X^{\oplus} h(x)\, d\mu_f(x) = \inf_x f(x) \odot h(x). \tag{2}$$

Theorem 1 is equivalent to the statement that all idempotent measures are absolutely continuous with respect to the standard idempotent measure $m_{\mathbb{1}}$.

4 Applications

4.1 Large deviation principle

The theory of large deviations concerned with the asymptotic estimation of probabilities of rare events, and typically provide exponential bound on probability of such events and characterize them. This theory has found many applications in information theory, coding theory, image processing, statistical mechanics, various kind of random processes (certain types of finite state Markov chains, Brownian motion, Wiener process), stochastic differential equations, etc. Contemporary large deviation theory uses various approaches.

Let Ω be a topological space and $\mathcal{A}$ be the algebra of its Borel sets. A family of probabilities (P_ε), $\varepsilon > 0$, on $(\Omega, \mathcal{A})$ obeys the *large deviation principle* if there exists a *rate function* $I : \Omega \to [0, \infty]$ such that

1) I is lower semi-continuous and $\Omega_a = \{\omega \in \Omega : I(\omega) \leq a\}$ is a compact set for any $a < \infty$,

2) $-\limsup_{\varepsilon \to 0} \varepsilon \log P_\varepsilon(C) \geq \inf_{\omega \in C} I(\omega)$ for each closed set $C \subset \Omega$,

3) $-\liminf_{\varepsilon \to 0} \varepsilon \log P_\varepsilon(U) \leq \inf_{\omega \in U} I(\omega)$ for each open set $U \subset \Omega$.

Then $m(A) = \inf_{\omega \in A} I(\omega)$ is a positive σ-$\oplus$-additive measure on $\mathcal{A}$. Therefore, it is naturally to generalize the previous definition in the following way [1, 23]. For any Borel set A let

$$P^{out} = \limsup_{\varepsilon \to 0} \varepsilon \log P_\varepsilon(A), \quad P^{in} = \liminf_{\varepsilon \to 0} \varepsilon \log P_\varepsilon(A).$$

One says that (P_ε) obeys *the weak large deviation principle*, if there exists a positive idempotent measure m on $(\Omega, \mathcal{A})$ such that

1) there exists a sequence (Ω_n) of compact subsets of Ω such that $m(\Omega_n^c) \to \mathbb{0} = +\infty$ as $n \to \infty$, where C^c stands for the complimentary set of C,

2) $m(C) \leq -P^{out}(C)$ for each closed $C \subset \Omega$,

2) $m(U) \geq -P^{in}(U)$ for each open $U \subset \Omega$.

Using Theorem 1 and its generalizations one can prove (see details in [1, 23]) that the large deviation principle and its weak version are actually equivalent for some (rather general) "good" spaces Ω. One can obtain also an interesting correspondence between the tightness conditions for probability and idempotent measures and for further generalizations on random sets see [14, 18]. Further generalization with respect to pseudo-additive measures are contained in [13].

4.2 Hamilton-Jacobi equation

The introduced generalized analysis is applied for solving nonlinear equations (ODE, PDE, difference equations, etc.) using now the pseudo-linear principle, which means that if u_1 and

u_2 are solutions of the considered nonlinear equation, then also $a_1 \odot u_1 \oplus a_2 \odot u_2$ is a solution for any constants a_1 and a_2 from $[a, b]$. This section is devoted to a short presentation of the theory of generalized solutions to the Cauchy problem

$$\begin{cases} \dfrac{\partial S}{\partial t} + H\left(x, \dfrac{\partial S}{\partial x}\right) = 0, \\ S|_{t=0} = S_0(x) \end{cases} \tag{3}$$

of a Hamilton-Jacobi equation (HJ) with Hamiltonian H being convex in p. Many important applications require nonsmooth Hamiltonian, e.g., in control theory. More details can be found in [9, 10, 11, 20]. Since any convex function can be written in the form

$$H(x, p) = \max_{u \in U}(pf(x, u) - g(x, u))$$

with some functions f, g, equation in (3) can be written in the equivalent form

$$\frac{\partial S}{\partial t} + \max_{u \in U}\left(\left\langle \frac{\partial S}{\partial x}, f(x, u)\right\rangle - g(x, u)\right) = 0,$$

which is called the *nonstationary Bellman differential equation* (HJB).

Let us discuss first, what difficulties occur when one tries to give a reasonable definition of the solutions to problem (3)? First, as simple examples shows, the classical (i.e., smooth) solution of the Cauchy problem (3) does not exist for large time even for smooth H and S_0. Therefore one cannot hope to obtain smooth solutions for nonsmooth H and S_0. On the other hand, in contrast with the theory of linear equations, where generalized solutions can be defined in the standard way as functionals on the space of test functions, there is no such approach in the theory of nonlinear equations.

The most popular approach to the theory of generalized solutions of the HJB equation is the vanishing viscosity method. However, this method cannot be used to construct a reasonable theory of generalized solutions to (3) for discontinuous initial functions. Furthermore, it is highly desirable to devise a theory of problems (3) on the basis of only intrinsic properties of HJB equations (i.e., regardless of the way in which the set of HJB equations is embedded in the set of higher-order equations). Such a theory, including solutions with discontinuous initial data, can be constructed for equations with convex Hamiltonians on the basis of idempotent analysis, using the new superposition principle for the solutions of (3) (which was first noted in [11]) and the idempotent analogue of the inner product

$$\langle f, g\rangle_P = \inf_x f(x) \odot g(x), \tag{4}$$

replacing the usual L_2-product.

There are many other nonlinear PDE treated by pseudo-analysis, see [6, 9, 10, 11, 12, 17, 22]. One of the main problem in the application on nonlinear PDE is the identification of operations $\oplus$ and $\odot$. Goard and Broadbridge [5] have obtained a close connection with Lie symmetry algebras. Namely, there are a number of computer added algorithms for finding Lie symmetry algebra and therefore this relation can be used for finding $\oplus$ and $\odot$.

4.3 Option pricing

The famous Black-Sholes (BS) and Cox-Ross-Rubinstein (CRR) formulas are basic results in the modern theory of option pricing in financial mathematics. They are usually deduced by means of stochastic analysis; various generalizations of these formulas were proposed using more sophisticated stochastic models for common stocks pricing evolution. The systematic deterministic approach to the option pricing leads to a different type of generalizations of BS and CRR formulas characterized by more rough assumptions on common stocks evolution (which are therefore easier to verify). This approach reduces the analysis of the option pricing to the study of certain homogeneous nonexpansive maps, which however are "strongly" infinite dimensional: they act on the spaces of functions defined on sets, which are not (even locally) compact. In the paper [8] there is obtained generalizations of the standard CRR and BS formulas can be obtained using the deterministic (actually game-theoretic) approach to option pricing and what class of homogeneous nonexpansive maps appear in these formulas, considering first a simplest model of financial market with only two securities in discrete time, then its generalization to the case of several common stocks, and then the continuous limit. The infinite dimensional generalization of the theory of homogeneous nonexpansive maps (which does not exists at the moment) would have direct applications to the analysis of derivative securities pricing. On the other hand, this approach, which uses neither martingales nor stochastic equations, makes the whole apparatus of the standard game theory appropriate for the study of option pricing.

References

[1] M. Akian, Densities of idempotent measures and large deviations, *Transactions of Amer. Math. Soc.* **351** (1999), 4515-4543.

[2] M. Akian, R. Bapat, S. Gaubert, Asymptotics of the Perron eigenvalue and eigenvector using max-algebra. *C.R. Acad. Sci. Paris* 327 (1)(1998), 927-932.

[3] R. A. Cuninghame-Green, *Minimax algebra.* Lecture Notes in Economics and Math. Systems 166, Berlin, Heidelberg, New–York, Springer–Verlag, 1979.

[4] D. Dubois, E. Pap, H. Prade, Hybrid probabilistic-possibilistic mixtures and utility functions, (Eds. J. Fodor,B. de Baets, P. Perny) Preferences and Decisions under Incomplete Knowledge , Springer-Verlag, 2000, 51-73.

[5] J. M. Goard, P. Broadbridge, Nonlinear Superposition Principles Obtained Lie Symmetry Methods. *J. Math. Anal. Appl.* **214** (1997), 633-657.

[6] J. Gunawardena (Ed.), *Idempotency*, Publications of the Newton Institute 11, CambridgeUniversity Press, Cambridge, 1998.

[7] E. P. Klement, R. Mesiar, E. Pap, *Triangular Norms.* Kluwer Academic Publishers, Dordrecht, 2000.

[8] V.N. Kolokoltsov, Nonexpansive maps and option pricing theory, *Kibernetika*, 34(6) (1998), 713-724.

[9] V.N. Kolokoltsov, V.P. Maslov, *Idempotent Analysis and its Applications*, Kluwer Academic Publ., 1998.

[10] G. L. Litvinov, V. P. Maslov (Eds.), *Idempotent Mathematics and Mathematical Physics*, Contemporary Mathematics 377, American Mathematical Society, Providence, Rhode Island, 2005.

[11] V. P. Maslov, *Asymptotic Methods for Solving Pseudodifferential Equations,* [in Russian], Nauka, Moscow, 1987.

[12] V. P. Maslov, S. N. Samborskii (Eds.), *Idempotent Analysis*, Number 13 in Advances in Soviet Mathematics, AMS, Providence, RI, 1992.

[13] Lj. Nedović, N. Ralević, T. Grbić, Large deviation principle with generated pseudo measures, *Fuzzy Sets and Systems* **155** (2005), 65-76.

[14] H.T. Nguyen, B. Bouchon-Meunier, *Random sets and large deviations principle as foundation for possibility measures*, Soft Computing 8 (2003), 61-70.

[15] E. Pap, Integral generated by decomposable measure. *Univ. u Novom Sadu Zb. Rad. Prirod.-Mat. Fak. Ser. Mat.* **20,1** (1990), 135-144.

[16] E. Pap, Null-Additive Set Functions, Dordrecht-Boston-London, Kluwer Academic Publishers, 1995.

[17] E. Pap, Pseudo-additive measures and their applications, (Ed. E. Pap) Handbook of Measure Theory, Elsevier, North-Holland, Amsterdam, 2002, 1237-1260.

[18] E. Pap, T. Grbić, Lj. Nedović, N. Ralević, Weak convergence of random sets, SISY 2005, 3rd Serbian-Hungarian Joint Symposium on Intelligent Systems, Subotica, 31. August-1. September 2005, 73-80.

[19] E. Pap, B. Mihailović: *A representation of a comonotone-$\check{\vee}$-additive and monotone functional by two Sugeno integrals*, Fuzzy Sets and Systems **155** (2005), 77-88.

[20] E. Pap, N. Ralević, Pseudo-Laplace Transform. *Nonlinear Analysis* **33**(1998), 533-550.

[21] E. Pap, I. Štajner, Generalized pseudo-convolution in the theory of probabilistic metric spaces, information, fuzzy numbers, optimization, system theory, *Fuzzy Sets and Systems* **102** (1999), 393-415.

[22] E. Pap, D. Vivona, Non-commutative and non-associative pseudo-analysis and its applications on nonlinear partial differential equations, *J. Math. Anal. Appl.* **246** (2000), 390-408.

[23] A. Puhalskii, *Large Deviations and Idempotent Probability*, CRC Press, 2001.

Technical Report
SCCH–TR–0535

Characterization of Kernels in the Frequency Domain
A Tutorial

Bernhard Moser
Software Competence Center Hagenberg
A-4232 Hagenberg, Austria
bernhard.moser@scch.at

Abstract — In this presentation we look on symmetric, positive definite functions from the point of view of the frequency domain. In the machine learning community these functions are referred to as kernels. Taking a spectral approach is quite instructive. First of all, translational and rotational invariant kernels can adequately be characterized by their spectral representation, and secondly, the dual concept of a regularization operator by means of an induced Green's function becomes more evident and handy.

Key words — *Kernels, Regularization, Reproducing Kernel Hilbert Space, Invariances*

Kompetenzzentren-Programm

1 Kernels

Definition 1. Let $\mathcal{X}$ be a non-empty set. A real-valued function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is said to be a positive definite kernel (short kernel) iff it is symmetric, that is, $k(x, y) = k(y, x)$ for all $x, y \in \mathcal{X}$, and positive definite, that is, $\sum_{i,j=1}^{n} c_i c_j k(x_i, x_j) \geq 0$ for any $n \in \mathbb{N}$ and choice of $x_1, \ldots, x_n \in \mathcal{X}$ and any choice of real numbers $c_1, \ldots, c_n \in \mathbb{R}$.

Remark 2. In contrary to linear algebra this definition of positive definiteness is common in the approximation and machine learning literature (compare [6, 7]).

Due to Aronszajn [1] kernels can be characterized by inner products.

Theorem 3. *For any kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, there exists a Hilbert space $\mathcal{H}$ and a mapping $\Phi : \mathcal{X} \to \mathcal{H}$ such that*

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle, \tag{1}$$

for any $x, y \in \mathcal{X}$, where $\langle ., . \rangle$ denotes the inner product in the Hilbert space.

Because of its relevance for kernel methods the property (1) in literature is sometimes chosen to be the starting point for the definition of a kernel (compare, e.g., [3]).

Theorem (3) does not tell how to construct the Hilbert space $\mathcal{H}$ (feature space) and the mapping Φ. Actually, $\mathcal{H}$ is not even uniquely determined.

2 Reproducing Kernel Hilbert Spaces

One way to obtain $\mathcal{H}$ is to start with $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}} := \{f : \mathcal{X} \to \mathbb{R}\}$, a set of real-valued functions on $\mathcal{X}$, and apply the Riesz representation theorem (cf, e.g., [5]), by which a bounded linear functional $\mathcal{F} : \mathcal{H} \to \mathbb{R}$, that is there is an upper bound $M > 0$ such that

$$\forall f \in \mathcal{H} : \mathcal{F}[f] \leq M \|f\|_{\mathcal{H}},$$

is uniquely represented by

$$\mathcal{F}[f] = \langle a, f \rangle_{\mathcal{H}} \tag{2}$$

for an element $a \in \mathcal{H}$ and where $\langle ., . \rangle_{\mathcal{H}}$ and $\|.\|_{\mathcal{H}}$ denotes the inner product of $\mathcal{H}$ and the norm induced by it, respectively.

If all evaluation functionals $\delta_x : \mathcal{H} \to \mathbb{R}$, $x \in \mathcal{X}$, given by

$$\delta_x[f] = f(x)$$

are postulated to be bounded, then due to Riesz representation theorem (2), to each element $x \in \mathcal{X}$ there is an element $K_x \in \mathcal{H}$ such that

$$f(x) = \delta_x[f] = \langle K_x, f \rangle_{\mathcal{H}}.$$

Particularly, for $f = K_y$ we obtain

$$K(x, y) := K_x(y) = \langle K_x, K_y \rangle_{\mathcal{H}} \tag{3}$$

This shows that each Hilbert space $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$, for which all evaluation functionals are bounded, induces a positive kernel K with feature map $\Phi(x) = K(x, .)$. Such Hilbert spaces are called *reproducing kernel Hilbert sapce* (*RKHS* for short) due to equation (3). Vice versa it can be shown that a positive kernel K induces uniquely a RKHS which is generated by K (see, e.g., [1, 6]).

3 Mercer Kernels

While the feature space $RKHS$ is a function space, Mercer's theorem demonstrates the construction of a feature space made up of sequences, that is ℓ_2, the set of square summable sequences (see [4]).

Theorem 4. *Suppose $k \in L_\infty(\mathcal{X}^2)$ is a symmetric real-valued function such that for all $f \in L_2(\mathcal{X})$ and any finite measure μ on $\mathcal{X}$, we have*

$$\int_{\mathcal{X}^2} k(x,y)f(x)f(y)d\mu(x)d\mu(y) \geq 0. \tag{4}$$

Let $\Psi_j \in L_2(\mathcal{X})$ be the normalized orthogonal eigenfunctions of the integral operator $T_{k,\mu}$ given by

$$T_{k,\mu} : L_2(\mathcal{X}) \to L_2(\mathcal{X})(T_{k,\mu})(x) := \int_{\mathcal{X}} k(x,y)f(y)d\mu(y) \tag{5}$$

associated with the eigenvalues $\lambda_j > 0$, sorted in non-decreasing order. Then

- $(\lambda_j)_j \in \ell_2$,
- $k(x,y) = \sum_{j=1}^{N} \lambda_j \Psi_j(x)\Psi_j(y)$ *holds for allmost all $(x,y) \in \mathcal{X}^2$. $N \in \mathbb{N}$, or $N = \infty$; in the latter case, the series converges absolutely and uniformly for allmost all $(x,y) \in \mathcal{X}^2$.*

4 Regularization

The following representer theorem is fundamental for optimization problems.

Theorem 5. *Let $\Omega : [0,\infty) \to \mathbb{R}$ be a strictly monotonically increasing function and*

$$c : ((\mathcal{X} \times \mathbb{R}^2)^m \to [0,\infty]$$

an arbitrary loss function, i.e., $c(x,y,f(x)) = 0$ iff $f(x) = y$. Then each minimizer of the regularized task

$$c((x_1,y_1,f(x_1)),\ldots,(x_m,y_m,f(x_m))) + \Omega(\|f\|_{\mathcal{H}})$$

admits a representation of the form

$$f(.) = \sum_i \alpha_i k(x_i,.).$$

Theorem 5 emphasizes the importance of the regularization term Ω which guarantees the optimization problem to be tractable. It also demonstrates the duality between kernels and regularization, which will be the issue next.

Definition 6. Let $\mathcal{R} : \mathcal{F} \to \mathcal{D}$, be a linear map from a dot product space $\mathcal{F} = \{f | f : \mathcal{X} \to \mathbb{R}\}$ into another dot product space $\mathcal{D}$. Then the regularization term is given by $\Omega[f] = \langle \mathcal{R}f, \mathcal{R}f \rangle$ and $\mathcal{R}$ is referred to as regularization operator.

Theorem 7. *For every positive definite linear self-adjoint operator $\hat{\mathcal{R}} : \mathcal{F} \rightarrow \mathcal{F}$ for which a Green's function exists, there is a corresponding $\mathcal{RKHS}$ $\mathcal{H}$ with reproducing kernel k, a dot product space $\mathcal{D}$ and a regularization operator $\mathcal{R} : \mathcal{F} \rightarrow \mathcal{D}$ such that*

$$f(x) = \langle \mathcal{R}k(x,.), \mathcal{R}f(.) \rangle_{\mathcal{D}}$$

The idea of the proof is the fact that the positive definite operator $\hat{\mathcal{R}}$ can be splitted in a product like $\hat{\mathcal{R}} = \mathcal{R}^*\mathcal{R}$ and that the Green's function of $G_x(.)$ of $\hat{\mathcal{R}}$ satisfies

$$f(x) = \langle \hat{\mathcal{R}}G_x(.), f(.) \rangle_{\mathcal{F}} = \langle \mathcal{R}G_x(.), \mathcal{R}f(.) \rangle_{\mathcal{D}}.$$

4.1 Regularization Opererator in the Frequency Domain

By Bochner's characterization theorem [2] for translational invariant kernels we get

$$k(x,y) = (2\pi)^{-\frac{N}{2}} \int_{\Omega} e^{i\langle \omega, (x-y) \rangle} v(\omega) d\omega. \tag{6}$$

The corresponding regularization operator to the kernel (7) turns out to be

$$\langle \mathcal{R}f, \mathcal{R}g \rangle_{\mathcal{D}} = (2\pi)^{\frac{N}{2}} \int_{\Omega} \frac{\overline{F[f](\omega)} F[g](\omega)}{v(\omega)} d\omega \tag{7}$$

where $F[f]$ denotes the Fourier transform of f.

4.2 Regularization Opererator for Mercer Kernels

For Mercer kernels there is an analogous result (see [6]). Let

$$k(x,y) = \sum_i \lambda_i \Psi_i(x) \Psi_i(y)$$

be a Mercer kernel, the corresponding regularization operator $\mathcal{R}$ is given by

$$(\mathcal{R}^*\mathcal{R})(x,y) = \sum_i \lambda_i^{-1} \Psi_i(x) \Psi_i(y)$$

provided the sum $\sum_i \lambda_i^{-1}$ is convergent.

References

[1] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

[2] S. Bochner. *Harmonic Analysis and the Theory of Probability*. University of California Press, Los Angeles, California, 1955.

[3] N. Cristianini and J. Shawe-Taylor. *Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[4] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, 209:415–446, 1909.

[5] W. Rudin. *Real and Complex Analysis*. McGraw-Hill, New York, 1966.

[6] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, 2002.

[7] G. Wahba. Soft and hard classification by reproducing kernel hilbert space methods. In *Proc. of the National Academy of Sciences of the United States of America*, volume 99(26), pages 16524–16530, 2002.

A Nonlinear Approximation Formula Generator for High Dimensional Data Based on Variable Selection and Genetic Programming

Werner Groißböck

Department of Knowledge-Based Mathematical Systems
Fuzzy Logic Laboratorium Linz-Hagenberg
Johannes Kepler University Linz, A-4040 Linz, Austria
werner.groissboeck@jku.at

Summary. A new approach for finding nonlinear approximation formulas for very high-dimensional data is presented. The method is based on linear regression, but instead of the original variables we use nonlinear terms with these variables. Such a formula is still linear in the parameters, so least squares can be applied to find the globally optimal parameters. We use an accelerated version of genetic programming to find the optimal nonlinear terms, and we use variable selection methods to select those terms leading to an approximation formula which shows an optimal balance of accuracy and simplicity. In general, evolutionary methods like genetic programming tend to produce many individuals with low fitness. To save computation time, an early stopping strategy in case of low fitness is used. The method was tested with two benchmark data sets (the Friedman data set in the KEEL repository http://sci2s.ugr.es/keel/, and the CPU data set in the UCI repository http://www.ics.uci.edu/ mlearn/MLRepository.html) where it produced approximation formulas which are more exact than the benchmark papers.

1 The approximation formula generator HDFormGen B

In this paper, the new algorithm *HDFormGen B* (A **For**mula **Gen**erator for **H**igh **D**imensional Data, Variant **B**)is introduced which is able to find an approximation formula with nonlinear terms for a high dimensional regression data set. With this algorithm, formulas similar to the following can been achieved:

$$\begin{aligned} y = & -413.505 \\ & + 2.3135 \cdot (P2offset \cdot Tgas) \\ & + 1.9948 \cdot ((P2offset + Tgas) - N) \\ & - 8.6768 \cdot (P1offset - \sin(Fm)) \\ & + 0.000000016942 \cdot ((CO2 \cdot P1offset) \cdot N) \end{aligned} \tag{1}$$

The new algorithm HDFormGen B is especially strong under the following conditions:

1. The higher the dimensionality, the better this algorithm fits. The reason is that in our algorithm no local fitting is performed, so the curse of dimensionality is not so bad. (For more details, see [9] , section 7.6.1 'Curse of Dimensionality'.)
2. If approximation formulas that are not very simple have to be detected, then this algorithm is the ideal choice. The crossover operator helps finding these formulas quite fast.
3. If the number of data points is not very high or the dimensionality is high, then local fitting is dangerous. So in these cases, our algorithm is a good choice.

The basic idea of the algorithm is the following:

- The structure of each of the nonlinear terms in the whole formula is found and optimized with the use of genetic programming (see [3] and [4]).
- The parameters of the formula can be optimized easily with a least squares algorithm. This can only be done, if the formula is linear in the parameters, so the genetic programming tool must not generate terms which contain additional parameters.

There is another aspect that has to be considered:
The terms that are used in the approximation formula finally shall be as uncorrelated to each other as possible. We want an approximation formula which is on the one hand as simple as possible, and on the other hand as exact as possible. So we have to find the most important nonlinear terms, such that the regression formula based on these terms is as good as possible. Variable selection methods like *forward selection* have been designed to fulfill this task. In *HDFormGen B* a variant of forward selection is used. For this reason, the basic concept of *forward selection* will be explained in the following rows:

- At first, the most important variable (or nonlinear term) is selected. This is that variable (or term) which is correlated strongest to the actual dependent y.
- Then the effects of the variables/terms selected so far are subtracted from the original dependent y. This is necessary to avoid that variables that are highly correlated to the first choice will be chosen again and again.
- Then, again the most important variable/term is selected.

- And again, the dependent is modified, such that the effects of the variables/terms chosen already are eliminated.
- Continue in this manner, until enough variables/terms are selected.

2 The new algorithm HDFormGen B

2.1 The core of the new algorithm

In the following, the original dependent is called y. At the beginning, the actual dependent is the original dependent $y_{actual} = y$. Later y_{actual} will be modified. The constant term $c = (1, \ldots, 1)^T$ is always the first variable that is chosen. But this variable is not counted as real variable. The algorithm performs the following steps:

1. An accelerated version of genetic programming (including a population of individuals and a crossover operator) is used to generate millions of very simple formulas. We select that formula x_A which is best correlated with the actual dependent y_{actual}. We look only at the absolute value of the correlation coefficient.
2. Then we modify y_{actual} such that all the parts of y that can be approximated with the regressors already chosen are subtracted, setting y_{actual} to $y - \hat{y}(c, x_A)$. Here $\hat{y}(c, x_A)$ is the linear best approximation of y with the use of the regressors c and x_A. We can say, y_{actual} is y made orthogonal to the regressors already chosen.
3. Once again the accelerated version of genetic programming is used to generate millions of very simple formulas. And now we select that formula x_B which is correlated strongest with the actual dependent y_{actual}. We look only at absolute values again.
4. Then once again, y_{actual} is made orthogonal to the regressors already chosen, so we set y_{actual} to $y - \hat{y}(c, x_A, x_B)$.
5. Continue in this manner, until a given number of regressor terms is selected or some other termination criterion is fulfilled.

2.2 The accelerated version of genetic programming - an overview

Stopping the calculation of the correlation coefficient as early as possible, when it can be seen that the checked individual is not worth spending additional time, accelerates the algorithm enormously. But how can this be carried out, if we have a population of individuals and not a single individual?

In the following lines the major steps of the accelerated genetic programming algorithm are described.

1. Generate an initial population with *popsize* individuals.

2. Evaluate each individual for $n1$ points of the training data set and estimate the correlation coefficient with the actual dependent by using only these $n1$ points.
3. Determine the $popsize_{small}$ best correlated individuals out of $popsize$, based on the estimated correlation coefficient. We look only at the absolute value of the correlation coefficient.
4. For these $popsize_{small}$ chosen individuals the *e*xact value of the fitness function (i.e. the absolute value of the correlation coefficient) using *a*ll the points of the training data set has to be calculated.
5. Produce a new generation of $popsize$ out of the $popsize_{small}$ chosen individuals:
   - Repeat the following, until we have enough new individuals. Choose randomly two of the $popsize_{small}$ individuals and compare their fitness. The better one is called the winner, and the other one is called the loser. Let the winner produce two offsprings, one is an exact copy of the winner, and the other offspring is made via crossover (as crossover partner, one of the $popsize_{small}$ individuals is chosen, which is neither the winner nor the loser).
   - The individual which is the best so far is always copied into the next generation ('elitism').
   - A small part of the new generation is produced in the same way as the initial population. This is one way of avoiding the problem with local optima. A mutation is not needed any more.
6. Go to step 2, until a termination criterion is fulfilled.

- As termination criterion, we usually take that a specific number of generations is reached.
- The parameter $popsize$ determines, how many individuals are evolved in the genetic programming algorithm. The parameter $popsize$ can take any positive integer number. The larger $popsize$ is, the more computation time is needed, and the better the results are. In our experiments, a $popsize$ of 5000 has been used successfully.
- The parameter $n1$ tells the algorithm, how many points are used to get a quick estimation of the correlation coefficient. $n1$ can be an arbitrary positive integer, but $n1$ shall not exceed the number of training data points. In our experiments, settings of $n1 = 30$, $n1 = 50$ and $n1 = 100$ have been used successfully.
- The parameter $popsize_{small}$ determines, how many individuals of the total population are selected to be examined in detail. The value of $popsize_{small}$ shall be much smaller than $popsize$, for example $popsize/10$.

3 Variants of the Formula Generator Algorithm Applied To Standard Benchmark Data Sets

To show the quality of our approximation formula generator, several standard data sets have been used. The main data source is the UCI repository, which can be found in the following address:

`http://www.ics.uci.edu/~mlearn/MLRepository.html`

The complete repository has been downloaded, and a few of the data sets have been selected for our experiments. The UCI repository is used by many software developers, so several reports concerning their results can be found. Nevertheless it has to be mentioned that many of the data sets in the UCI repository can not be used for examining our methods. The vast majority of the data sets is a mixture of numerical and non-numerical data. Many other data sets have missing values. And many data sets have not more than 2 variables. All these data sets can not be used here. We can only use data sets that can be handled with multidimensional regression.

3.1 The data set auto-mpg

The first data set is called 'auto-mpg'. This data can be found in the directory 'auto-mpg' of the UCI-repository. The data concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of 3 multivalued discrete and 5 continuous attributes.
Number of instances: 398
Number of attributes: 9
The data set contains the following attributes:

mpg: continuous; the dependent variable
cylinders: multi-valued discrete
displacement: continuous
horsepower: continuous
weight: continuous
acceleration: continuous
model year: multi-valued discrete
origin: multi-valued discrete
car name: string (unique for each instance)

In the original data set, the attribute *horsepower* has six missing values. So we deleted the corresponding rows in our data set. And we deleted the attribute *carname*, because our algorithm cannot handle strings. So finally we have a data set with 392 rows and 8 columns.

The original data set has been split into two parts. We have taken the first `90%` of the data as training-data, and the rest as test-data. Then we

made an approximation formula for 'mpg' with our algorithm. And finally we calculated the mean absolute error (=MAE) for the test data.

One of the best formulas that we reach with our algorithms is the following:

$$\begin{aligned} mpg =& 7.7974 \\ & - 0.0084204 \cdot ((sqrtabs(weight) - modelyear)modelyear) \\ & + 1.6747 \cdot (\sin(sqrtabs((horsepower + cylinders)))) \\ & - 0.88459 \cdot (\cos((modelyear + modelyear)) - origin) \end{aligned} \tag{2}$$

This formula is quite simple, and it leads to an MAE of 2.6861 for the test data. We found another formula with an MAE that is even slightly lower, namely 2.5182, but the corresponding formula is a bit more complicated:

$$\begin{aligned} mpg =& 35.6145 \\ & - 0.67801 \cdot (sqrtabs(weight) - (modelyear + origin)) \\ & + 1.5074 \cdot (\sin(sqrtabs((horsepower + sqrtabs(horsepower))))) \\ & + 34.605 \cdot (\sin(\cos(sqrtabs((acceleration + modelyear))))) \\ & - 0.062681 \cdot (acceleration \cdot \cos((modelyear + modelyear))) \\ & - 0.99804 \cdot (\cos((acceleration \cdot \cos(cylinders)))) \end{aligned} \tag{3}$$

In the UCI-Repository we found a benchmark paper see [11] that uses various methods to find an approximation for mpg. The situation is not exactly the same, because in the paper mentioned the `10%` test data have been chosen in a different way. Still we want to mention that the best method in [11], reached an MAE of 2.02. And the worst method in [11] reached an MAE of 6.53. So our formulas are comparable to these methods. For more details see [11].

3.2 The data set cpu

The data set 'cpu' can be found in the directory 'cpu-performance' of the UCI-repository.
Number of instances: 209
Number of attributes: 10
The data set contains the following attributes:

vendor name: string
model name: string
MYCT: machine cycle time in nanoseconds (integer)
MMIN: minimum main memory in kilobytes (integer)
MMAX: maximum main memory in kilobytes (integer)

CACH: cache memory in kilobytes (integer)
CHMIN: minimum channels in units (integer)
CHMAX: maximum channels in units (integer)
PRP: published relative performance (integer); the dependent variable;
ERP: estimated relative performance via linear regression (integer)

At first we deleted the attributes *vendorname* and *modelname* because our algorithm can not handle strings. Furthermore the data set contains the attribute ERP, which is an old estimation for PRP. So we have to delete the attribute ERP, because we do not want to generate an approximation formula by using the results of an old approximation. This would be too easy. So finally we have only 7 attributes remaining. Before the core of our algorithm has been run, we split the data into two parts: `70%` of the 209 instances have been randomly chosen to play the role of the training data. And the other `30%` play the role of the test-data. This has been done similar to one of our benchmark papers - see [7]. In [7] the process of selecting the training data randomly was repeated 20 times to get more exact results. We do not do this here, too, because then we would have to run our evolution based algorithm 20 times, and this costs too much computational effort.

Our algorithm has been started 10 times. Roughly 3.7 seconds are necessary per term for performing the evolutionary part of the algorithm. Totally we received ten approximation formulas, with an average MAE of 23.33 determined for the test data set. The worst MAE is only 25.15, and the best MAE is 23.06. The best formula is the following:

$$\begin{aligned} PRP =& 16.344 \\ &+ 0.0032443 \cdot (sqrtabs((MMIN \cdot (MMAX \cdot CHMAX)))) \\ &+ 0.7936 \cdot ((CACH - CHMAX) - \sin(CHMAX)) \end{aligned} \tag{4}$$

In our benchmark paper (see [7]), various different methods have been tried out. The best method leads to an MAE of 38.0. So compared to this paper, our method leads to a more exact approximation.

In the paper [11], an MAE of 28.1 has been reached. It has to be mentioned, that in [11], the experiment is not exactly the same, as far as the process of splitting the data into training data and test data is concerned. For more details, see [11].

3.3 The data set friedman

The data set 'friedman' can be found in the KEEL repository, which is located in

http://sci2s.ugr.es/keel/

In the keel repository, benchmark papers can be found. For the friedman data set, a quite actual (2004) benchmark paper is available (see [5]).

We try to design our experiments as similar as possible to the benchmark paper, to get comparable results. In the benchmark paper, the following is done:

'This is a synthetic benchmark data set. Each sample consists of five inputs and one output. The formula for the data generation is $t = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4^4 + 5x_5^5 + \varepsilon$, where ε is a Gaussian random noise $N(0,1)$, and $x_1, ..., x_5$ are uniformly distributed over the domain $[0,1]$. 1400 samples were created, of which 200 samples were randomly chosen for network training and 200 samples for validation. The remaining 1000 samples were used for network testing.'

In the KEEL repository, the data sets are already available as described in [5]. So we have a 200 sample training data set, and a 200 sample validation data set, and a 1000 sample test data set. Unlike the benchmark paper, we do not need any validation data. So we only take the 200 sample training data set to find an approximation formula, and we take the 1000 sample test data set to determine the quality. As a quality measure, here the MSE is used, according to the benchmark data.

The best formula that we get is the following:

$$\begin{aligned} salida =& 4.8843 \\ &+ 10.1761 \cdot (entrada4 + \sin((entrada2 \cdot (entrada1 + (entrada1 + entrada1))))) \\ &- 5.3183 \cdot (\sin((entrada3 + (entrada3 + entrada3))) - entrada5) \end{aligned} \tag{5}$$

The MAE of this formula is 0.889281, and the MSE of this formula is 1.23629 . In the benchmark-paper (see [5]), the best method leads to an MSE of 4.502 . So our formula is much more exact.

References

1. Hastie, T., Tibshirani, R., and Friedman J. , "The Elements of Statistical Learning: Data Mining, Inference, and Prediction.", Springer Berlin, 2001
2. F. E. Harrell jr., "Regression modeling strategies: With applications to linear models, logistic regression and survival analysis", Springer Series in Statistics, 2001
3. J. R. Koza, "Genetic Programming", The MIT Press, Cambridge, Massachusetts, 1992
4. J. R. Koza, "Genetic Programming II", The MIT Press, Cambridge, Massachusetts, 1994

5. W. M. Lee, C. P. Lim, K. K. Yuen, S. M. Lo, "A Hybrid Neural Network Model for Noisy Data Regression", IEEE Transactions on Systems, Man and Cybernetics, Part B 34:2, Pages 951-960, 2004
6. L. Ljung, "System Identification: Theory for the User", ISBN 0-13-656695-2, Prentice Hall PTR, New Jersey, 1999
7. Merz, C. J., Pazzani, M. J., "Combining Neural Network Regression Estimates with Regularized Linear Weights", Advances in Neural Information Processing Systems 9, edited by M.C. Mozer, M.I. Jordan, and T. Petsche, 1997
8. A. Miller, "Subset Selection in Regression - Second Edition", ISBN 1-58488-171-2 Chapman & Hall/CRC Boca Raton London New York Washington, D.C. 2002
9. O. Nelles, "Nonlinear System Identification - From Classical Approaches to Neural Networks and Fuzzy Models", ISBN 3-540-67369-5 Springer-Verlag Berlin Heidelberg New York, 2001
10. W. H. Press, S. A. Teukolsky, W. T. Vetterling and P.B. Flannery, "Numerical Recipes in C: The Art of Scientific Computing", Cambridge University Press, Cambridge, U.K., second ed., 1992
11. J. R. Quinlan, "Combining Instance-Based and Model-Based Learning", Proceedings on the Tenth International Conference of Machine Learning, Pages 236-243, University of Massachusetts, Morgan Kaufmann, 1993