Studies in Computational Intelligence 445

Christian Moewes
Andreas Nürnberger (Eds.)

Computational Intelligence in Intelligent Data Analysis

Springer

Studies in Computational Intelligence 445

Editor-in-Chief

Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

For further volumes:
http://www.springer.com/series/7092

Christian Moewes and Andreas Nürnberger (Eds.)

Computational Intelligence in Intelligent Data Analysis

Springer

Editors
Christian Moewes
Faculty of Computer Science
Otto-von-Guericke University Magdeburg
Magdeburg
Germany

Andreas Nürnberger
Faculty of Computer Science
Otto-von-Guericke University Magdeburg
Magdeburg
Germany

ISSN 1860-949X
e-ISSN 1860-9503
ISBN 978-3-642-32377-5
e-ISBN 978-3-642-32378-2
DOI 10.1007/978-3-642-32378-2
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012943592

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Complex systems and their phenomena are ubiquitous as they can be found in biology, finance, the humanities, management sciences, medicine, physics and similar fields. For many problems in these fields, there are no conventional ways to mathematically or analytically solve them completely at low cost. On the other hand, nature already solved many optimization problems efficiently. Computational intelligence attempts to mimic nature-inspired problem-solving strategies and methods. These strategies can be used to study, model and analyze complex systems such that it becomes feasible to handle them. Key areas of computational intelligence are artificial neural networks, evolutionary computation and fuzzy systems. This volume contains a collection of papers dealing with computational intelligence in data analysis.

Nowadays, as more and more data storage gets affordable with increasing bandwidth and decreasing production costs of RAM or sensors, the amount of data is rapidly growing, even faster than Moore predicted it for the speed of computers. This is especially the case for complex systems as they are data intensive. Storing the data only, however, does not solve real-world problems we might face. In fact, the more data is stored, the harder it is for human beings to see useful regularities or patterns. Therefore the research field of *data analysis* has been created and developed beautiful tools to find patterns in a huge amount of data. When these methods are furthermore enhanced by human ideas and concepts that ensure interpretability, soundness and applicability, we talk about *intelligent data analysis*.

As only a few researchers in that field, Rudolf Kruse has contributed in many important ways to the understanding, modeling and application of computational intelligence methods. In l996 he was appointed to a full professor position in the Faculty of Computer Science at the Otto-von-Guericke University Magdeburg, Germany. There he is the head of the computational intelligence group. Rudolf Kruse has co-authored more than 30 books and 330 technical papers. He has the competence to develop beautiful theories which are typically applied to solve real-world problems.

Recently, the research of Rudolf Kruse and his collaborators enabled new directions in the fields of data mining and intelligent data analysis. Of particular importance are his contributions to mainly four research areas. These are fuzzy data analysis, hybrid intelligent systems, uncertainty in knowledge-based systems, and intelligent data analysis. On occasion of his 60th birthday, we collected 20 original papers in this volume of leading researchers in the field of computational intelligence. Among the authors are many former doctorates that were mentored by Rudolf Kruse and also a couple of his research partners he collaborated with over many years.

First of all, we express our gratitude to Janusz Kacprzyk who suggested to compose this book. Janusz actually made it possible to publish it in the Springer series Studies in Computational Intelligence.

Furthermore we are very thankful to all authors who accepted our invitation to contribute a chapter to this volume. We are very grateful to all reviewers who helped to improved the papers. Last not least, we thank the Springer-Verlag for the excellent collaboration in publishing this book in time.

Magdeburg
September 2012

Christian Moewes
Andreas Nürnberger

Contents

Part III: Uncertainty in Knowledge-Based Systems

Part IV: Intelligent Data Analysis

Part V: Applications

Part VI: Closing Remarks

Part I
Fuzzy Data Analysis

Objective Functions for Fuzzy Clustering

Christian Borgelt

Abstract. Fuzzy clustering comprises a family of prototype-based clustering methods that can be formulated as the problem of minimizing an objective function. These methods can be seen as "fuzzifications" of, for example, the classical c-means algorithm, which strives to minimize the sum of the (squared) distances between the data points and the cluster centers to which they are assigned. However, it is well known that in order to "fuzzify" such a crisp clustering approach, it is not enough to merely allow values from the unit interval for the variables encoding the assignments of the data points to the clusters (that is, for the elements of the partition matrix): the minimum is still obtained for a crisp data point assignment. As a consequence, additional means have to be employed in the objective function in order to obtain actual degrees of membership. This paper surveys the most common fuzzification means and examines and compares their properties.

1 Introduction

The general objective of *clustering* or *cluster analysis* [14, 23, 26, 21] is to group given objects in such a way that objects from the same cluster are as similar as possible, while objects from different clusters are as dissimilar as possible. In order to formalize the notion of similarity, so that it becomes mathematically treatable, it is usually expressed as a *distance measure* between points (or vectors) representing the objects in a metric space, usually $\mathbb{R}^m$. Two objects are then seen as the more similar, the smaller the distance between the data points that represent them.

A common approach to describe the clusters is to use *prototypes* that capture the location and possibly also the shape and size of the clusters in the data space. With such an approach the general objective of clustering can be reformulated as the task

Christian Borgelt
European Centre for Soft Computing, Edificio de Investigación,
33600 Mieres (Asturias), Spain
e-mail: christian@borgelt.net

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 3–16.
springerlink.com

to find a set of cluster prototypes together with an assignment of the data points to them, so that the data points are as close as possible to their assigned prototypes. By formalizing this approach, and using for the prototypes only points in the data space that represent the *cluster centers*, one obtains immediately the objective function of classical c-means clustering [1, 19, 32]: simply sum the (squared) distances of the data points to the center of the cluster to which they are assigned. The c-means algorithm then strives to minimize this objective function.

Unfortunately, c-means clustering always partitions the data, that is, each data point is assigned to one cluster and one cluster only. This is often inappropriate, as it can lead to somewhat arbitrary cluster boundaries and certainly does not treat points properly that lie between two (or more) clusters without belonging to any of them unambiguously. Solutions to this problem consist in either using a probabilistic approach, like applying the expectation maximization (EM) algorithm to a mixture of Gaussians (see, for example, [11, 15, 6]), or to employ one of the different "fuzzifications" of the classical crisp scheme (see, for instance, [37, 13, 2, 5, 21, 7]).

In this paper I focus on the latter approach, that is, on how the objective function of classical c-means clustering can be modified in order to obtain graded cluster memberships. I survey different methods that have been suggested in the literature and examine and compare their properties. The remainder of this paper is organized as follows: Section 2 introduces the presuppositions made and the notation used in this paper. Section 3 briefly reviews the formal basis of the classical c-means algorithm. The following two sections discuss the main classes of "fuzzification" approaches: Section 4 explores membership transformation and Section 5 examines membership regularization as tools to obtain graded memberships from a modified objective function. Finally, Section 6 draws conclusions from the discussion.

2 Presuppositions and Notation

We are given a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with n data points, each of which is an m-dimensional real-valued vector, that is, $\forall j; 1 \leq j \leq n : \mathbf{x}_j = (x_{j1}, \ldots, x_{jm}) \in \mathbb{R}^m$. These data points are to be grouped into c clusters, each of which is described by a prototype $\mathbf{c}_i$, $i = 1, \ldots, c$. The set of all prototypes is denoted by $\mathbf{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_c\}$. I confine myself here to cluster prototypes that consist merely of a cluster center, that is, $\forall i; 1 \leq i \leq c : \mathbf{c}_i = (c_{i1}, \ldots, c_{im}) \in \mathbb{R}^m$. The assignment of the data points to the cluster centers is encoded as a $c \times n$ matrix $\mathbf{U} = (u_{ij})_{1 \leq i \leq c; 1 \leq j \leq n}$, which is often called the *partition matrix*. In the crisp case, a matrix element $u_{ij} \in \{0, 1\}$ states whether data point $\mathbf{x}_j$ belongs to cluster $\mathbf{c}_i$ or not. In the fuzzy case, $u_{ij} \in [0, 1]$ states the degree to which $\mathbf{x}_j$ belongs to $\mathbf{c}_i$ (degree of membership).

In this paper I also confine myself to the (squared) Euclidean distance as the measure for the distance between a data point $\mathbf{x}_j$ and a cluster center $\mathbf{c}_i$, that is,

$$d_{ij}^2 \;=\; d^2(\mathbf{c}_i, \mathbf{x}_j) \;=\; (\mathbf{x}_j - \mathbf{c}_i)^\top (\mathbf{x}_j - \mathbf{c}_i) \;=\; \sum_{k=1}^{m} (x_{jk} - c_{ik})^2.$$

A common alternative is the (squared) Mahalanobis distance with a cluster specific covariance matrix Σ_i [18, 17], that is, $d_{ij}^2 = (\mathbf{x}_j - \mathbf{c}_i)^\top \Sigma_i^{-1} (\mathbf{x}_j - \mathbf{c}_i)$. However, this choice adds at least a shape parameter and in some approaches also a size parameter to the cluster prototypes (see, for example, [5, 21, 7]). Nevertheless, extending the approaches to this distance measure is usually fairly straightforward. An extension to the L_1-distance [24], that is, to $d_{ij} = \sum_{k=1}^{m} |x_{jk} - c_{ik}|$, or to other Minkowski metrics is less simple to achieve, but certainly beyond the scope of this paper.

3 Classical c-Means Clustering

As already stated, classical c-means clustering strives to find, for a given data set $\mathbf{X}$, a set $\mathbf{C}$ of cluster centers and a partition matrix $\mathbf{U}$, such that the objective function

$$J(\mathbf{X}, \mathbf{C}, \mathbf{U}) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij} d_{ij}^2$$

is minimized under the constraints $\forall i; 1 \leq i \leq c : \forall j; 1 \leq j \leq n : u_{ij} \in \{0, 1\}$ and $\forall j; 1 \leq j \leq n : \sum_{i=1}^{c} u_{ij} = 1$. These constraints ensure that each data point is assigned to one cluster and to one cluster only (crisp partition of the data set).

Since the minimum cannot be found directly using analytical means, an *alternating optimization* scheme is employed. At the beginning the cluster centers are initialized randomly, for example, by selecting c data points arbitrarily or by sampling c points from some distribution on the data space. Then the two steps of *partition matrix update* (data point assignment) and *cluster center update* are iterated until convergence, that is, until the cluster centers do not change anymore.

In the partition matrix update each data point $\mathbf{x}_j$ is assigned to the cluster $\mathbf{c}_i$, the center of which is closest to it, that is, the partition matrix is updated according to

$$u_{ij} = \begin{cases} 1, \text{ if } i = \operatorname{argmin}_{i=1}^{c} d_{ij}^2, \\ 0, \text{ otherwise.} \end{cases}$$

In the cluster center update each cluster center is recomputed as the mean of the data points that were assigned to it (hence the name c-means clustering), that is,

$$\mathbf{c}_i = \frac{\sum_{j=1}^{n} u_{ij} \mathbf{x}_j^2}{\sum_{j=1}^{n} u_{ij}}.$$

This update process is guaranteed to converge and usually does so after fairly few steps. However, it is fairly sensitive to the initial conditions (i.e. the initial cluster centers), due to which it can yield undesired results, which are caused by local minima of the objective function. In order to handle this drawback, it is usually recommended to execute the clustering algorithm multiple times and take the best result, that is, the result that yields the smallest value of the objective function.

In order to obtain *degrees of membership*, it may seem, at first sight, to be sufficient to simply extend the allowed range of values of the u_{ij} from the set $\{0,1\}$ to the real interval $[0,1]$, but to make no changes to the objective function itself. However, this is not the case: the optimum of the objective function is obtained for a crisp assignment, regardless of whether we enforce a crisp assignment or not.

This can easily be demonstrated as follows: let $k_j = \operatorname{argmin}_{i=1}^{c} d_{ij}^2$, that is, let k_j be the index of the cluster center closest to the data point $\mathbf{x}_j$. Then it is

$$\begin{aligned} J(\mathbf{X},\mathbf{C},\mathbf{U}) &= \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}\, d_{ij}^2 \;\geq\; \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}\, d_{k_j j}^2 \;=\; \sum_{j=1}^{n} d_{k_j j}^2 \underbrace{\sum_{i=1}^{c} u_{ij}}_{=1 \text{ (due to the constraints)}} \\ &= \sum_{j=1}^{n} \left(1 \cdot d_{k_j j}^2 + \sum_{\substack{i=1 \\ i \neq k_j}}^{c} 0 \cdot d_{ij}^2 \right). \end{aligned}$$

Therefore it is best to set $\forall j; 1 \leq j \leq n : u_{k_j j} = 1$ and $u_{ij} = 0$ for $1 \leq i \leq c$, $i \neq k_j$. In other words: the objective function is minimized by assigning each data point crisply to the closest cluster, even though we allowed for degrees of membership.

4 Fuzzification by Membership Transformation

Since we cannot obtain degrees of membership by merely expanding the range of values of the u_{ij}, we have to modify the objective function if we desire graded assignments. The most common approach is to apply a transformation to the membership degrees, that is, to use an objective function of the form

$$J(\mathbf{X},\mathbf{C},\mathbf{U}) = \sum_{i=1}^{c}\sum_{j=1}^{n} h(u_{ij})\, d_{ij}^2,$$

where h is a convex function on the real interval $[0,1]$. This general form was first studied in [27], where the convexity of h was derived as follows: for simplicity, we confine ourselves to two clusters $\mathbf{c}_1$ and $\mathbf{c}_2$ and consider the terms of the objective function that refer to a single data point $\mathbf{x}_j$. That is, we consider $J(\mathbf{x}_j,\mathbf{c}_1,\mathbf{c}_2,u_{1j},u_{2j}) = h(u_{1j})\, d_{1j}^2 + h(u_{2j})\, d_{2j}^2$ and study how it behaves for different values u_{1j} and u_{2j}. Note that a crisp assignment should not be ruled out categorically, namely if the distances d_{1j} and d_{2j} differ significantly. Hence we assume that d_{1j} and d_{2j} differ only slightly, so that a graded assignment is actually desired.

$J(\mathbf{x}_j,\mathbf{c}_1,\mathbf{c}_2,u_{1j},u_{2j})$ is minimized by choosing u_{1j} and u_{2j} appropriately. Exploiting $\sum_{i=1}^{c} u_{ij} = 1$ yields $J(\mathbf{x}_j,\mathbf{c}_1,\mathbf{c}_2,u_{1j}) = h(u_{1j})\, d_{1j}^2 + h(1-u_{1j})\, d_{2j}^2$. A necessary condition for a minimum is $\frac{\partial}{\partial u_{1j}} J(\mathbf{x}_j,\mathbf{c}_1,\mathbf{c}_2,u_{1j}) = h'(u_{1j})\, d_{1j}^2 - h'(1-u_{1j})\, d_{2j}^2 = 0$, where $'$ denotes taking the derivative w.r.t. the argument of the function. This leads to $h'(u_{1j})\, d_{1j}^2 = h'(1-u_{1j})\, d_{2j}^2$, which yields another argument that a graded assignment cannot be optimal without any function h: if h is the identity, we have $h'(u_{1j}) = h'(1-u_{1j}) = 1$ and thus the equation cannot hold if the distances differ.

For the further analysis let us assume, without loss of generality, that $d_{1j} < d_{2j}$, which implies $h'(u_{1j}) > h'(1-u_{1j})$. In addition, we know that $u_{1j} > u_{2j} = 1-u_{1j}$, because the degree of membership should be higher for the cluster that is closer. In other words, the function h must be the steeper, the greater its argument. Therefore it must be a convex function on the unit interval [27].

Since we confine ourselves to the Euclidean distance (see Section 2), we can already derive the update rule for the cluster centers, namely by exploiting that a necessary condition for a minimum of the objective function J is that the partial derivatives w.r.t. the cluster centers vanish. Therefore we have $\forall k; 1 \leq k \leq c$:

$$\nabla_{\mathbf{c}_k} J(\mathbf{X}, \mathbf{C}, \mathbf{U}) = \nabla_{\mathbf{c}_k} \sum_{i=1}^{c} \sum_{j=1}^{n} h(u_{ij}) (\mathbf{x}_j - \mathbf{c}_i)^\top (\mathbf{x}_j - \mathbf{c}_i) = -2 \sum_{j=1}^{n} h(u_{ij}) (\mathbf{x}_j - \mathbf{c}_i) \stackrel{!}{=} 0.$$

Independent of the function h, it follows immediately

$$\mathbf{c}_i = \frac{\sum_{j=1}^{n} h(u_{ij}) \mathbf{x}_j}{\sum_{j=1}^{n} h(u_{ij})}.$$

This update rule already shows one of the core drawbacks of a fuzzification by membership transformation, namely that the transformation function enters the update of the cluster centers. It would be more intuitive to use the membership degrees directly as the weights for the mean computation, which would also ensure that all data points enter with the same total unit weight (since $\sum_{i=1}^{c} u_{ij} = 1$ by definition). However, the weights are rather the transformed membership degrees $h(u_{ij})$, which gives unequal weight to the data points as they need not sum to 1.

It may be argued, though, that this effect can actually be desirable: due to the convexity of the function h the total weight $\sum_{i=1}^{c} h(u_{ij})$ of data points $\mathbf{x}_j$ with a less ambiguous assignment is higher than that of more ambiguously assigned data points. Hence in this scheme the locations of the cluster centers depend more strongly on the data points that are "typical" for the clusters. Such an effect is very much in the spirit of, for instance, robust regression techniques, in which data points receive a lower weight if they do not fit well to the regression function. This connection to robust statistical methods was explored in more detail, for example, in [10].

In order to derive the update rule for the partition matrix (and thus for the membership degrees u_{ij}) we need to know the exact form of the function h. The most common choice is $h(u_{ij}) = u_{ij}^2$, which leads to the standard objective function of fuzzy clustering [13]. The more general form $h(u_{ij}) = u_{ij}^w$ was introduced in [2]. The exponent w, $w > 1$, is called the *fuzzifier*, since it controls the "fuzziness" of the data point assignments: the higher w, the softer the boundaries between the clusters. This leads to the commonly used objective function [2, 5, 21, 7]

$$J(\mathbf{X}, \mathbf{U}, \mathbf{C}) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^w d_{ij}^2.$$

The update rule for the membership degrees is now derived by incorporating the constraints $\forall j; 1 \leq j \leq n : \sum_{i=1}^{c} u_{ij} = 1$ with Lagrange multipliers into the objective function. This yields the Lagrange function

$$L(\mathbf{X}, \mathbf{U}, \mathbf{C}, \Lambda) = \underbrace{\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} d_{ij}^{2}}_{=J(\mathbf{X},\mathbf{U},\mathbf{C})} + \sum_{j=1}^{n} \lambda_j \left(1 - \sum_{i=1}^{c} u_{ij}\right),$$

where $\Lambda = (\lambda_1, \ldots, \lambda_n)$ are the Lagrange multipliers, one per constraint.

Since a necessary condition for a minimum of the Lagrange function is that the partial derivatives w.r.t. the membership degrees vanish, we obtain

$$\frac{\partial}{\partial u_{kl}} L(\mathbf{X}, \mathbf{U}, \mathbf{C}, \Lambda) = w\, u_{kl}^{w-1}\, d_{kl}^{2} - \lambda_l \stackrel{!}{=} 0 \qquad \text{and thus} \qquad u_{kl} = \left(\frac{\lambda_l}{w\, d_{kl}^{2}}\right)^{\frac{1}{w-1}}.$$

Summing these equations over the clusters (in order to be able to exploit the corresponding constraints on the membership degrees, which are recovered from the fact that it is a necessary condition for a minimum that the partial derivatives of the Lagrange function w.r.t. the Lagrange multipliers vanish), we get

$$1 = \sum_{i=1}^{c} u_{ij} = \sum_{i=1}^{c} \left(\frac{\lambda_j}{w\, d_{ij}^{2}}\right)^{\frac{1}{w-1}} \qquad \text{and thus} \qquad \lambda_j = \left(\sum_{i=1}^{c} \left(w\, d_{ij}^{2}\right)^{\frac{1}{1-w}}\right)^{1-w}.$$

Therefore we finally have for the membership degrees $\forall i; 1 \leq i \leq c$: $\forall j; 1 \leq j \leq n$:

$$u_{ij} = \frac{d_{ij}^{\frac{2}{1-w}}}{\sum_{k=1}^{c} d_{kj}^{\frac{2}{1-w}}} \qquad \text{and thus for } w = 2: \qquad u_{ij} = \frac{d_{ij}^{-2}}{\sum_{k=1}^{c} d_{kj}^{-2}}.$$

This rule is fairly intuitive, as it updates the membership degrees according to the relative inverse squared distances of the data points to the cluster centers.

However, this rule also has the disadvantage that it necessarily yields a graded assignment. Regardless of how far a data point is from a cluster center, it will always receive a non-vanishing degree of membership to the corresponding cluster. The undesirable results that can be caused by this property in the presence of clusters with fairly uneven numbers of members have been demonstrated clearly in [27].

In addition, it was revealed in [27] that the reason lies essentially in the fact that $h'(u_{ij}) = \frac{\mathrm{d}}{\mathrm{d}u_{ij}} u_{ij}^{w} = w u_{ij}^{w-1}$ vanishes at $u_{ij} = 0$. This suggests the idea to use a transformation function that does not have this property and thus allows, at least for sufficiently large distance relationships, a crisp assignment of data points to cluster centers. In [27] the function $h(u_{ij}) = \alpha u_{ij}^{2} + (1-\alpha) u_{ij}$, $\alpha \in (0, 1]$, or, with a more easily interpretable parametrization, $h(u_{ij}) = \frac{1-\beta}{1+\beta} u_{ij}^{2} + \frac{2\beta}{1+\beta} u_{ij}$, $\beta \in [0, 1)$, was suggested as such a transformation. It relies on the standard function $h(u_{ij}) = u_{ij}^{2}$ and mixes it with the identity to avoid a vanishing derivative at zero. The parameter β is,

for two clusters, the ratio of the smaller to the larger squared distance, at and below which we get a crisp assignment [27]. It therefore takes the place of the fuzzifier w: the smaller β, the softer the boundaries between the clusters.

The update rule for the membership degrees is derived in essentially the same way as for $h(u_{ij}) = u_{ij}^w$, although one has to pay attention to the fact that crisp assignments are now possible and thus some membership degrees may vanish. The detailed derivation, which I omit here, can be found in [27] or in [7]. It yields

$$u_{ij} = \frac{u'_{ij}}{\sum_{k=1}^{c} u'_{kj}} \qquad \text{with} \qquad u'_{ij} = \max\left\{0,\, d_{ij}^{-2} - \frac{\beta}{1+\beta(c_j-1)} \sum_{k=1}^{c_j} d_{\varsigma(k)j}^{-2}\right\},$$

where $\varsigma : \{1,\ldots,c\} \to \{1,\ldots c\}$ is a mapping function for the cluster indices such that $\forall i; 1 \le i < c : d_{\varsigma(i)j} \le d_{\varsigma(i+1)j}$ (that is, ς sorts the distances ascendingly) and

$$c_j = \max\left\{k \;\middle|\; d_{\varsigma(k)j}^{-2} > \frac{\beta}{1+\beta(k-1)} \sum_{i=1}^{k} d_{\varsigma(i)j}^{-2}\right\}$$

is the number of clusters to which the data point $\mathbf{x}_j$ has a non-vanishing membership. This update rule is fairly interpretable, as it still assigns membership degrees essentially according to the relative inverse squared distances to the clusters, but subtracts an offset from them, which makes crisp assignments possible.

5 Fuzzification by Membership Regularization

We have seen that transforming the membership degrees in the objective function has the disadvantage that the transformation function appears in the update rule for the cluster centers. In order to avoid this drawback, one may try to achieve a fuzzification by leaving the membership degrees in their weighting of the (squared) distances untouched. Graded memberships are rather achieved by adding a regularization term to the objective function, which pushes the minimum away from a crisp assignment. Most commonly, the objective function then takes the form

$$J(\mathbf{X},\mathbf{C},\mathbf{U}) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij} d_{ij}^2 + \gamma \sum_{i=1}^{c}\sum_{j=1}^{n} f(u_{ij}),$$

where f is a convex function on the real interval $[0,1]$. The parameter γ takes the place of the fuzzifier w: the higher γ, the softer the boundaries between the clusters.

To analyze this objective function, we use the same basic means as in the preceding section: we confine ourselves to two clusters $\mathbf{c}_1$ and $\mathbf{c}_2$ and consider the terms of the objective function that refer to a single data point $\mathbf{x}_j$, that is, we consider $J(\mathbf{x}_j,\mathbf{c}_1,\mathbf{c}_2,u_{1j},u_{2j}) = u_{1j}d_{1j}^2 + u_{2j}d_{2j}^2 + \gamma f(u_{1j}) + \gamma f(u_{2j})$. Since $u_{2j} = 1-u_{1j}$, it is $J(\mathbf{x}_j,\mathbf{c}_1,\mathbf{c}_2,u_{1j}) = u_{1j}d_{1j}^2 + (1-u_{1j})d_{2j}^2 + \gamma f(u_{1j}) + \gamma f(1-u_{1j})$. A necessary condition for a minimum is $\frac{\partial}{\partial u_{1j}} J(\mathbf{x}_j,\mathbf{c}_1,\mathbf{c}_2,u_{1j}) = d_{1j}^2 - d_{2j}^2 + \gamma f'(u_{1j}) - \gamma f'(1-u_{1j}) = 0$,

where $'$ denotes taking the derivative w.r.t. the argument of the function. This leads to the simple condition $d_{1j}^2 + \gamma f'(u_{1j}) = d_{2j}^2 + \gamma f'(1-u_{1j})$.

We now assume again, without loss of generality, that $d_{1j} < d_{2j}$, which implies $f'(u_{1j}) > f'(1-u_{1j})$. In addition we know $u_{1j} > u_{2j} = 1-u_{1j}$, because the degree of membership should be higher for the cluster that is closer. In other words, the function f must be the steeper, the greater its argument. Hence it must be a convex function on the unit interval in order to allow for graded memberships.

More concretely, we obtain $(d_{2j}^2 - d_{1j}^2)/\gamma = f'(u_{1j}) - f'(1-u_{1j})$ as a condition for a minimum. Since f is a convex function on the unit interval, the maximum value of the right hand side is $f'(1) - f'(0)$. If $f'(1) - f'(0) < \infty$, we have the possibility of crisp assignments, because in this case there exist values for d_{1j}^2, d_{2j}^2 and γ such that the minimum of the function $J(\mathbf{x}_j, \mathbf{c}_1, \mathbf{c}_2, u_{1j})$ w.r.t. u_{ij} either does not exist or lies outside the unit interval. In such a situation the best choice is the crisp assignment $u_{1j} = 1$ and $u_{2j} = 0$ (still assuming that $d_{1j} < d_{2j}$).

To obtain the update rule for the cluster centers we can simply transfer the result from the preceding section, since the regularization term does not refer to the cluster centers. Therefore we have the simple rule (because here $h(u_{ij}) = u_{ij}$)

$$\mathbf{c}_i = \frac{\sum_{j=1}^{n} u_{ij}\mathbf{x}_j}{\sum_{j=1}^{n} u_{ij}}.$$

This demonstrates the advantage of a membership regularization approach, because the membership degrees are directly the weights with which the data points enter the mean computation that yields the new cluster center.

In order to derive the update rule for the membership degrees, we have to respect the constraints $\forall j; 1 \leq j \leq n : \sum_{i=1}^{c} u_{ij} = 1$. This is achieved in the usual way (cf. the preceding section) by incorporating them with Lagrange multipliers into the objective function. The resulting Lagrange function is

$$L(\mathbf{X}, \mathbf{U}, \mathbf{C}, \Lambda) = \underbrace{\sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij} d_{ij}^2 + \gamma \sum_{i=1}^{c}\sum_{j=1}^{n} f(u_{ij})}_{=J(\mathbf{X},\mathbf{C},\mathbf{U})} + \sum_{j=1}^{n} \lambda_j \left(1 - \sum_{i=1}^{c} u_{ij}\right),$$

where $\Lambda = (\lambda_1, \ldots, \lambda_n)$ are the Lagrange multipliers, one per constraint.

Since a necessary condition for a minimum of the Lagrange function is that the partial derivatives w.r.t. the membership degrees vanish, we obtain

$$\frac{\partial}{\partial u_{kl}} L(\mathbf{X}, \mathbf{U}, \mathbf{C}) = d_{kl}^2 + \gamma f'(u_{kl}) - \lambda_l \stackrel{!}{=} 0 \qquad \text{and thus} \qquad u_{kl} = f'^{-1}\left(\frac{\lambda_l - d_{kl}^2}{\gamma}\right),$$

where $'$ denotes taking the derivative w.r.t. the argument of the function and f'^{-1} denotes the inverse of the derivative of the function f. In analogy to Section 4, the constraints on the membership degrees are now exploited to obtain $1 = \sum_{k=1}^{c} u_{kj} = \sum_{k=1}^{c} f'^{-1}((\lambda_j - d_{kj}^2)/\gamma)$. This equation has to be solved for λ_j and the result has

to be used to substitute λ_l in the expression for the u_{kl} derived above. However, in order to do so, we need to know the exact form of the regularization function f.

The regularization functions f that have been suggested in the literature (concrete examples are studied below) can be seen as derived from a maximum entropy approach. That is, the term of the objective function that forces the u_{ij} to minimize the weighted sum of squared distances is complemented by a term that forces them to maximize the entropies of the distributions over the clusters, the u_{ij} describe for each data point. Thus the u_{ij} are pushed away from a crisp assignment, which has minimum entropy. Generally, such an approach starts from the objective function

$$J(\mathbf{X},\mathbf{C},\mathbf{U}) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}d_{ij}^2 - \gamma\sum_{j=1}^{n} H(\mathbf{u}_j),$$

where $\mathbf{u}_j = (u_{1j},\ldots,u_{cj})$ comprises the degrees of membership the data point $\mathbf{x}_j$ has to the different clusters. H computes their entropy, as $\mathbf{u}_j$ is, at least formally, a probability distribution, since it satisfies $\forall i; 1 \le i \le c : u_{ij} \in [0,1]$ and $\sum_{i=1}^{c} u_{ij} = 1$.

In order to develop the maximum entropy approach in more detail, we consider the generalized entropy proposed by Daróczy in [9]. Let $\mathbf{p} = (p_1,\ldots,p_r)$ be a probability distribution over r values. Then *Daróczy entropy* is defined as

$$H_\beta(\mathbf{p}) = \frac{2^{\beta-1}}{2^{\beta-1}-1}\sum_{i=1}^{r} p_i(1-p_i^{\beta-1}) = \frac{2^{\beta-1}}{2^{\beta-1}-1}\left(1-\sum_{i=1}^{r} p_i^{\beta}\right).$$

From this general formula the well-known *Shannon entropy* [38] can be derived as

$$H_1(\mathbf{p}) = \lim_{\beta\to 1} H_\beta(\mathbf{p}) = -\sum_{i=1}^{r} p_i \log_2 p_i.$$

Employing it in the entropy-regularized objective function leads to

$$J(\mathbf{X},\mathbf{C},\mathbf{U}) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}d_{ij}^2 + \gamma\sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}\ln u_{ij},$$

where the factor $1/\ln 2$ (which stems from the relation $\log_2 u_{ij} = \ln u_{ij}/\ln 2$) is incorporated into the factor γ, as the natural logarithm allows for easier mathematical treatment. That is, we have $f(u_{ij}) = u_{ij}\ln u_{ij}$ [25, 31, 33, 8] and therefore obtain $f'(u_{ij}) = 1+\ln u_{ij}$ and $f'^{-1}(y) = e^{y-1}$. Using the latter in the formulas obtained above for deriving the update rule for the membership degrees yields

$$u_{ij} = \frac{e^{-d_{ij}^2/\gamma}}{\sum_{k=1}^{c} e^{-d_{kj}^2/\gamma}}.$$

As was pointed out in [35, 20], this update rule relates the approach very closely to the expectation maximization (EM) algorithm for Gaussian mixtures [11, 15, 6], since by setting $\gamma = 2\sigma^2$, we obtain exactly the formula for the expectation step. As a

consequence, this update rule can be interpreted as computing the probability that a data point $\mathbf{x}_j$ was sampled from a Gaussian distribution centered at $\mathbf{c}_i$ and having the variance σ^2. In addition, since the update rule for the cluster centers coincides with the maximization step, this form of fuzzy clustering is actually indistinguishable from the expectation maximization algorithm for a mixture of Gaussians.

It should be noted that $f'(u_{ij}) = 1 + \ln u_{ij}$ implies $f'(1) - f'(0) = \infty$ and thus Shannon entropy regularization always yields graded assignments. However, this drawback is less harmful here, because $e^{-d_{ij}^2/\gamma}$ is much "steeper" than d_{ij}^{-2} and thus is less prone to produce undesired results (cf. also the discussion in [12]).

Another commonly used special case of Daróczy entropy is so-called *quadratic entropy*, which results if we set the parameter $\beta = 2$, that is,

$$H_2(\mathbf{p}) = 2\sum_{i=1}^{r} p_i(1-p_i) = 2 - 2\sum_{i=1}^{r} p_i^2.$$

Employing it in the entropy-regularized objective function leads to

$$J(\mathbf{X},\mathbf{C},\mathbf{U}) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}d_{ij}^2 + \gamma\sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^2,$$

as the constant term 2 has no influence on the location of the minimum and thus can be discarded, and the factor 2 can be incorporated into the factor γ. That is, we have $f(u_{ij}) = u_{ij}^2$ [34] and therefore obtain $f'(u_{ij}) = 2u_{ij}$ and $f'^{-1}(y) = \frac{y}{2}$.

In order to derive the update rule for the memberships, one has to pay attention to the fact that $f'(1) - f'(0) = 2$. Therefore crisp assignments are possible and some membership degrees may vanish. However, the detailed derivation can easily be found by following, for example, the same lines as for the analogous approach in the preceding section, which also allowed for vanishing membership degrees.

The resulting membership degree update rule is $\forall i : 1 \le i \le c : \forall j : 1 \le j \le n :$

$$u_{ij} = \max\left\{0, \frac{1}{c_j}\left(1 + \sum_{k=1}^{c_j}\frac{d_{\varsigma(k)j}^2}{2\gamma}\right) - \frac{d_{ij}}{2\gamma}\right\},$$

where $\varsigma : \{1,\ldots,c\} \to \{1,\ldots c\}$ is a mapping function for the cluster indices such that $\forall i; 1 \le i < c : d_{\varsigma(i)j} \le d_{\varsigma(i+1)j}$ (that is, ς sorts the distances ascendingly) and

$$c_j = \max\left\{k \;\middle|\; \sum_{i=1}^{k} d_{\varsigma(i)j}^2 > k\, d_{kj} - 2\gamma\right\}$$

is the number of clusters to which the data point $\mathbf{x}_j$ has a non-vanishing membership. In this update rule 2γ can be interpreted as a reference distance relative to which all distances are judged. For two clusters, 2γ is the difference between the distances of a data point to the cluster centers, at and above which a crisp assignment is used. Clearly, this is equivalent to saying that the distances, if measured in 2γ units, must differ by less than 1 in order to obtain a graded assignment.

A disadvantage of this update rule is that it refers to the difference of the distances rather than their ratio, which seems more intuitive. As a consequence, a data point that has distance x to one cluster and distance y to the other is assigned in exactly the same way as a data point that has distance $x+z$ to the first cluster and distance $y+z$ to the second, regardless of the value of z (provided $z \geq -\min\{x,y\}$).

Alternatives to the discussed approaches modified the Shannon entropy term, using, for instance, $f(u_{ij}) = u_{ij}\ln u_{ij} + (1-u_{ij})\ln(1-u_{ij})$ [42], or replaced it with Kullback-Leibler information divergence [30] to the (estimated) cluster probability distribution [22], that is, $f(u_{ij}) = u_{ij}\ln\frac{u_{ij}}{p_i}$ with $p_i = \frac{1}{n}\sum_{j=1}^{n} u_{ij}$.

It has also been tried to use $f(u_{ij}) = u_{ij}^w$ [41, 36], but combined with $h(u_{ij}) = u_{ij}^w$ (to avoid technical complications), so that the objective function is effectively

$$J(\mathbf{X},\mathbf{C},\mathbf{U}) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^w (d_{ij}^2 + \gamma).$$

Hence this is actually a hybrid approach that combines membership transformation and regularization. Another hybrid approach, proposed in [40], combines $h(u_{ij}) = u_{ij}^w$ and Shannon entropy regularization $f(u_{ij}) = u_{ij}\ln u_{ij}$. Finally, a generalized objective function was presented in [3] and analyzed in more detail in [43].

It should be noted, though, that the approach of [16], which is covered by the generalized objective function of [3] and based on

$$J(\mathbf{X},\mathbf{C},\mathbf{U}) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^w d_{ij}^2 - \gamma\sum_{i=1}^{c} p_i^2 \qquad \text{with} \qquad p_i = \frac{1}{n}\sum_{j=1}^{n} u_{ij},$$

is *not* a membership regularization scheme, as it yields crisp assignments unless $w > 1$. In this approach the entropy term (which is added rather than subtracted) serves the purpose to choose the number of clusters automatically.

A closely related approach is *possibilistic clustering* [28, 29], which eliminates the constraints $\forall j; 1 \leq j \leq n : \sum_{i=1}^{c} u_{ij} = 1$ and is based on the objective function

$$J(\mathbf{X},\mathbf{C},\mathbf{U}) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^w d_{ij}^2 + \sum_{i=1}^{c}\eta_i\sum_{j=1}^{n}(1-u_{ij})^w.$$

Here the η_i are suitable positive numbers (one per cluster $\mathbf{c}_i$, $1 \leq i \leq c$) that determine the distance at which the membership degree of a point to a cluster is 0.5. They are usually initialized, based on the result of a preceding run of standard fuzzy clustering, as the average fuzzy intra-cluster distance $\eta_i = \sum_{j=1}^{n} u_{ij}^w d_{ij}^2 / \sum_{j=1}^{n} u_{ij}^w$ and may or may not be updated in each iteration [28].

Although this approach is useful in certain applications, it should be noted that the objective function of possibilistic clustering is truly optimized only if all clusters are identical [39], because the missing constraints decouple the clusters. Thus it actually *requires* that the optimization process gets stuck in a local optimum in order to yield useful results, which is a somewhat strange property.

6 Conclusions

Since classical c-means clustering does not yield graded data point assignments, even if one allows the membership variables to take values in the unit interval, the objective function has to be modified if graded assignments are desired. There are two fundamental approaches to this: transforming the membership degrees or adding a membership regularization term. In both cases variants can be derived that allow partially crisp assignments, that is, allow for vanishing membership degrees, as well as variants that enforce graded assignments regardless of the data. All of these variants have advantages and disadvantages: membership transformation suffers generally from the fact that the transformation function enters the cluster center update, but uses a fairly intuitive relative inverse squared distance scheme for the membership updates. Quadratic entropy regularization allows for vanishing membership degrees, but refers to distance differences rather than more intuitive distance ratios. Shannon entropy regularization leads to a procedure that is equivalent to the expectation maximization (EM) algorithm for a mixture of Gaussian and thus is not a specifically "fuzzy" approach anymore. However, judging from the discussion in [12] due to which the forced graded assignment is unproblematic, its practical advantages make it, in my personal opinion, the most recommendable approach.

References

[1] Ball, G.H., Hall, D.J.: A clustering technique for summarizing multivariate data. Behavioral Science 12(2), 153–155 (1967)
[2] Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)
[3] Bezdek, J.C., Hathaway, R.J.: Visual cluster validity (VCV) displays for prototype generator clustering methods. In: Proc. 12th IEEE Int. Conf. on Fuzzy Systems, FUZZ-IEEE 2003, Saint Louis, MO, vol. 2, pp. 875–880. IEEE Press, Piscataway (2003)
[4] Bezdek, J.C., Pal, N.: Fuzzy Models for Pattern Recognition. IEEE Press, New York (1992)
[5] Bezdek, J.C., Keller, J., Krishnapuram, R., Pal, N.: Fuzzy Models and Algorithms for Pattern Recognition and Image Processing. Kluwer, Dordrecht (1999)
[6] Bilmes, J.: A gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Tech. Rep. ICSI-TR-97-021, University of Berkeley, CA, USA (1997)
[7] Borgelt, C.: Prototype-based classification and clustering. Habilitationsschrift, Otto-von-Guericke University of Magdeburg, Germany (2005)
[8] Boujemaa, N.: Generalized competitive clustering for image segmentation. In: Proc. 19th Int. Meeting North American Fuzzy Information Processing Society, NAFIPS 2000, Atlanta, GA, pp. 133–137. IEEE Press, Piscataway (2000)
[9] Daróczy, Z.: Generalized information functions. Information and Control 16(1), 36–51 (1970)
[10] Davé, R.N., Krishnapuram, R.: Robust clustering methods: A unified view. IEEE Trans on Fuzzy Systems 5(1997), 270–293 (1997)
[11] Dempster, A.P., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society (Series B) 39, 1–38 (1977)

[12] Döring, C., Borgelt, C., Kruse, R.: Effects of irrelevant attributes in fuzzy clustering. In: Proc. 14th IEEE Int. Conf. on Fuzzy Systems, FUZZ-IEEE 2005, Reno, NV, pp. 862–866. IEEE Press, Piscataway (2005)
[13] Dunn, J.C.: A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. Journal of Cybernetics 3(3), 32–57 (1973); reprinted in [4], 82–101
[14] Everitt, B.S.: Cluster Analysis. Heinemann, London (1981)
[15] Everitt, B.S., Hand, D.J.: Finite Mixture Distributions. Chapman & Hall, London (1981)
[16] Frigui, H., Krishnapuram, R.: Clustering by competitive agglomeration. Pattern Recognition 30(7), 1109–1119 (1997)
[17] Gath, I., Geva, A.B.: Unsupervised optimal fuzzy clustering. IEEE Trans Pattern Analysis and Machine Intelligence (PAMI) 11, 773–781 (1989); reprinted in [4], 211–218
[18] Gustafson, E.E., Kessel, W.C.: Fuzzy clustering with a fuzzy covariance matrix. In: Proc. of the IEEE Conf. on Decision and Control, CDC 1979, San Diego, CA, pp. 761–766. IEEE Press, Piscataway (1979); reprinted in [4], 117–122
[19] Hartigan, J.A., Wong, M.A.: A *k*-means clustering algorithm. Applied Statistics 28, 100–108 (1979)
[20] Honda, K., Ichihashi, H.: Regularized linear fuzzy clustering and probabilistic PCA mixture models. IEEE Trans Fuzzy Systems 13(4), 508–516 (2005)
[21] Höppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy Cluster Analysis. John Wiley & Sons, Ltd., Chichester (1999)
[22] Ichihashi, H., Miyagishi, K., Honda, K.: Fuzzy c-means clustering with regularization by K-L information. In: Proc. 10th IEEE Int. Conf. on Fuzzy Systems, FUZZ-IEEE 2001, Melbourne, Australia, pp. 924–927. IEEE Press, Piscataway (2001)
[23] Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs (1988)
[24] Jajuga, K.: l_1-norm based fuzzy clustering. Fuzzy Sets and Systems 39(1), 43–50 (2003)
[25] Karayiannis, N.B.: MECA: maximum entropy clustering algorithm. In: Proc. 3rd IEEE Int. Conf. on Fuzzy Systems, FUZZ-IEEE 1994, Orlando, FL, vol. I, pp. 630–635. IEEE Press, Piscataway (1994)
[26] Kaufman, L., Rousseeuw, P.: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons, Ltd., New York (1990)
[27] Klawonn, F., Höppner, F.: What Is Fuzzy about Fuzzy Clustering? Understanding and Improving the Concept of the Fuzzifier. In: Berthold, M., Lenz, H.-J., Bradley, E., Kruse, R., Borgelt, C. (eds.) IDA 2003. LNCS, vol. 2810, pp. 254–264. Springer, Heidelberg (2003)
[28] Krishnapuram, R., Keller, J.M.: A possibilistic approach to clustering. IEEE Trans on Fuzzy Systems 1(2), 98–110 (1993)
[29] Krishnapuram, R., Keller, J.M.: The possibilistic *c*-means algorithm: Insights and recommendations. IEEE Trans on Fuzzy Systems 4(3), 385–393 (1996)
[30] Kullback, S., Leibler, R.A.: On information and sufficiency. Annals of Mathematical Statistics 22, 79–86 (1951)
[31] Li, R.P., Mukaidono, M.: A maximum entropy approach to fuzzy clustering. In: Proc. 4th IEEE Int. Conf. on Fuzzy Systems, FUZZ-IEEE 1994, Yokohama, Japan, pp. 2227–2232. IEEE Press, Piscataway (1995)
[32] Lloyd, S.: Least squares quantization in PCM. IEEE Trans Information Theory 28, 129–137 (1982)

[33] Miyamoto, S., Mukaidono, M.: Fuzzy c-means as a regularization and maximum entropy approach. In: Proc. 7th Int. Fuzzy Systems Association World Congress, IFSA 1997, Prague, Czech Republic, vol. II, pp. 86–92 (1997)
[34] Miyamoto, S., Umayahara, K.: Fuzzy clustering by quadratic regularization. In: Proc. IEEE Int. Conf. on Fuzzy Systems/IEEE World Congress on Computational Intelligence, WCCI 1998, Anchorage, AK, vol. 2, pp. 1394–1399. IEEE Press, Piscataway (1998)
[35] Mori, Y., Honda, K., Kanda, A., Ichihashi, H.: A unified view of probabilistic PCA and regularized linear fuzzy clustering. In: Proc. Int. Joint Conf. on Neural Networks, IJCNN 2003, Portland, OR, pp. 541–546. IEEE Press, Piscataway (2003)
[36] Özdemir, D., Akarun, L.: A fuzzy algorithm for color quantization of images. Pattern Recognition 35, 1785–1791 (2002)
[37] Ruspini, E.H.: A new approach to clustering. Information and Control 15(1), 22–32 (1969); reprinted in [4], 63–70
[38] Shannon, C.E.: The mathematical theory of communication. The Bell System Technical Journal 27, 379–423 (1948)
[39] Timm, H., Borgelt, C., Döring, C., Kruse, R.: An extension to possibilistic fuzzy cluster analysis. Fuzzy Sets and Systems 147, 3–16 (2004)
[40] Wei, C., Fahn, C.: The multisynapse neural network and its application to fuzzy clustering. IEEE Trans Neural Networks 13(3), 600–618 (2002)
[41] Yang, M.S.: On a class of fuzzy classification maximum likelihood procedures. Fuzzy Sets and Systems 57, 365–375 (2004)
[42] Yasuda, M., Furuhashi, T., Matsuzaki, M., Okuma, S.: Fuzzy clustering using deterministic annealing method and its statistical mechanical characteristics. In: Proc. 10th IEEE Int. Conf. on Fuzzy Systems, FUZZ-IEEE 2001, Melbourne, Australia, vol. 2, pp. 797–800. IEEE Press, Piscataway (2001)
[43] Yu, J., Yang, M.S.: A generalized fuzzy clustering regularization model with optimality tests and model complexity analysis. IEEE Trans Fuzzy Systems 15(5), 904–915 (2007)

Efficient Learning of Classifiers Based on the 2-Additive Choquet Integral*

Eyke Hüllermeier and Ali Fallah Tehrani

Abstract. In a recent work, we proposed a generalization of logistic regression based on the Choquet integral. Our approach, referred to as *choquistic regression*, makes it possible to capture non-linear dependencies and interactions among predictor variables while preserving two important properties of logistic regression, namely the comprehensibility of the model and the possibility to ensure its monotonicity in individual predictors. Unsurprisingly, these benefits come at the expense of an increased computational complexity of the underlying maximum likelihood estimation. In this paper, we propose two approaches for reducing this complexity in the specific though practically relevant case of the 2-additive Choquet integral. Apart from theoretical results, we also present an experimental study in which we compare the two variants with the original implementation of choquistic regression.

1 Introduction

The Choquet integral is well-known as a flexible aggregation function and, as such, has been used in various fields of application [14, 11, 21]. In machine learning, it is less common so far, although the interest in using the Choquet integral as a mathematical tool for tackling problems like classification, regression and ranking is increasing [12, 13, 22, 1, 2, 9].

In [8], we proposed a method called "choquistic regression", which is a generalization of logistic regression based on the Choquet integral. Choquistic regression has a number of appealing properties. Most notably, it combines three features in a non-trivial way, namely monotonicity, nonlinearity and interpretability. As for the

Eyke Hüllermeier · Ali Fallah Tehrani
Department of Mathematics and Computer Science, University of Marburg,
35032 Marburg, German
e-mail: {eyke,fallah}@mathematik.uni-marburg.de

* Dedicated to Professor Rudolf Kruse on the occasion of his 60^{th} birthday.

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 17–29.
springerlink.com

first, a monotone dependence between the input and output attributes is often desirable in a classification setting and sometimes even requested by the application [3, 19, 10]. At the same time, the Choquet integral also allows for modeling interactions between different attributes in a flexible, nonlinear way. Last but not least, thanks to the existence of natural measures for quantifying the influence of individual (e.g., the Shapley value) and the interaction between groups of features (e.g., the interaction index), it provides important insights into the model, thereby supporting interpretability [7].

Compared to standard logistic regression, these benefits are coming at the expense of an increased computational complexity of the underlying learning algorithm, which solves a maximum likelihood estimation problem. This is mainly caused by the large number of parameters of the fuzzy measure on which the Choquet integral is based, and the complicated dependency between these parameters. In this paper, we propose two approaches for reducing this complexity in the specific though practically relevant case of the 2-additive Choquet integral. To this end, we shall try to optimally exploit the simplified structure of a 2-additive measure in comparison to a non-additive measure in the general case.

The rest of this paper is organized as follows. In the next section, we briefly recall the basic definition of the (discrete) Choquet integral and some related notions. In Section 3, we sketch the idea of using the Choquet integral for binary classification and recall the basics of choquistic regression. In Section 4, we develop two alternative formulations of the learning (likelihood maximization) problem, both pursuing the same goal of complexity reduction. In Section 5, we present an experimental study in which we compare the two variants with the original implementation of choquistic regression, prior to concluding the paper with a few remarks in Section 6.

2 The Discrete Choquet Integral

In this section, we start with a brief recapitulation of the (discrete) Choquet integral and, along the way, introduce the main mathematical notation used throughout the paper.

Let $C = \{c_1, \ldots, c_m\}$ be a finite set and $\mu : 2^C \to [0,1]$ a measure. For each $A \subseteq C$, the value $\mu(A)$ can be interpreted as the weight or, say, the importance of the set of elements A. A standard assumption on a measure $\mu(\cdot)$, which is, for example, at the core of probability theory, is additivity: $\mu(A \cup B) = \mu(A) + \mu(B)$ for all $A, B \subseteq C$ such that $A \cap B = \emptyset$. Unfortunately, additive measures cannot model any kind of interaction between elements: Extending a set of elements A by a set of elements B always increases the weight $\mu(A)$ by the weight $\mu(B)$, regardless of the "context" A.

This lack of expressivity motivates the use of non-additive measures, also called capacities or fuzzy measures, which are simply normalized and monotone but not necessarily additive [20]:

$$\begin{aligned} &\mu(\emptyset) = 0, \mu(C) = 1 \\ &\mu(A) \leq \mu(B) \text{ for all } A \subseteq B \subseteq C \end{aligned} \tag{1}$$

A useful representation of non-additive measures, that we shall explore later on for learning Choquet integrals, is in terms of the *Möbius transform*:

$$\mu(B) = \sum_{A \subseteq B} m_\mu(A) \tag{2}$$

for all $B \subseteq C$, where the Möbius transform m_μ of the measure μ is defined as follows:

$$m_\mu(A) = \sum_{B \subseteq A} (-1)^{|A|-|B|} \mu(B) \,. \tag{3}$$

A measure μ is said to be k-order additive, or simply k-additive, if k is the smallest integer such that $m(A) = 0$ for all $A \subseteq C$ with $|A| > k$. This property is interesting for several reasons. In particular, as can be seen from (2), it means that a measure μ can formally be specified by significantly fewer than 2^m values, which are needed in the general case.

Suppose the "criteria" $c_i \in C$ are simply considered as binary features, which are either present or absent in a set A. Mathematically, $\mu(A)$ can then also be seen as an *integral* of the indicator function of A, namely the function f_A given by $f_A(c) = 1$ if $c \in A$ and $= 0$ otherwise. Now, suppose that $f : C \to \mathbb{R}_+$ is any non-negative function that assigns a *value* to each criterion c_i; for example, $f(c_i)$ might be the degree to which a candidate satisfies criterion c_i. An important question, then, is how to *aggregate* the evaluations of individual criteria, i.e., the values $f(c_i)$, into an overall evaluation, in which the criteria are properly weighted according to the measure μ. Mathematically, this overall evaluation can be considered as an integral $\mathscr{C}_\mu(f)$ of the function f with respect to the measure μ.

Indeed, if μ is an additive measure, the standard integral just corresponds to the *weighted mean*

$$\mathscr{C}_\mu(f) = \sum_{i=1}^{m} w_i \cdot f(c_i) = \sum_{i=1}^{m} \mu(\{c_i\}) \cdot f(c_i) \,, \tag{4}$$

which is a natural aggregation operator in this case. A non-trivial question, however, is how to generalize (4) in the case where μ is non-additive.

This question, namely how to define the integral of a function with respect to a non-additive measure (not necessarily restricted to the discrete case), is answered in a satisfactory way by the Choquet integral, which has first been proposed for additive measures by Vitali [23] and later on for non-additive measures by Choquet [4]. In the discrete case, the Choquet integral is formally defined as follows:

$$\mathscr{C}_\mu(f) = \sum_{i=1}^{m} \Big(f(c_{(i)}) - f(c_{(i-1)}) \Big) \cdot \mu\left(A_{(i)}\right) \,,$$

where $(\cdot)$ is a permutation of $\{1,\ldots,m\}$ such that $0 \leq f(c_{(1)}) \leq f(c_{(2)}) \leq \ldots \leq f(c_{(m)})$ (and $f(c_{(0)}) = 0$ by definition), and $A_{(i)} = \{c_{(i)},\ldots,c_{(m)}\}$. In terms of the Möbius transform of μ, the Choquet integral can also be expressed as follows:

$$\mathscr{C}_\mu(f) = \sum_{T \subseteq C} m(T) \cdot \min_{i \in T} f(c_i) \tag{5}$$

where $T_{(i)} = \big\{S \cup \{c_{(i)}\} \,|\, S \subset \{c_{(i+1)}, \ldots, c_{(m)}\}\big\}$.

3 The Choquet Integral as a Tool for Classification

As mentioned earlier, the Choquet integral has been used as a tool for different types of machine learning problems. In the following, we focus on the setting of binary classification, where the goal is to predict the value of an output (response) variable $y \in \mathscr{Y} = \{0,1\}$ for a given instance represented in terms of a feature vector

$$x = (x_1, \ldots, x_m) \in \mathscr{X} = \mathscr{X}_1 \times \mathscr{X}_2 \times \ldots \times \mathscr{X}_m$$

More specifically, the goal is to learn a classifier $\mathscr{L} : \mathscr{X} \to \mathscr{Y}$ from a given set of (independent and identically distributed) training data

$$\mathscr{D} = \Big\{ (x^{(i)}, y^{(i)}) \Big\}_{i=1}^{n} \subset (\mathscr{X} \times \mathscr{Y})^n \tag{6}$$

so as to minimize the risk

$$R(\mathscr{L}) = \int_{\mathscr{X} \times \mathscr{Y}} \ell(\mathscr{L}(x), y) \, d\mathbf{P}_{XY}(x,y) \,, \tag{7}$$

where $\ell(\cdot)$ is a loss function (e.g., the simple 0/1 loss given by $\ell(\hat{y}, y) = 0$ if $\hat{y} = y$ and $= 1$ if $\hat{y} \neq y$).

In this context, the predictor variables (features) play the role of the criteria $c_i \in C$. The Choquet integral can be used in order to model nonlinear dependencies between these variables and the response, thus taking interactions between predictors into account while preserving monotonicity in each individual feature. This can be done in different ways. In the following, we propose a model that can be seen as an extension of logistic regression.

3.1 Choquistic Regression

The key idea of the method of "choquistic regression" as proposed in [8] is to model the log-odds ratio between the positive $(y = 1)$ and the negative $(y = 0)$ class as a function of the Choquet integral of the input attributes; thus, the affine function $x \mapsto w_0 + w^\top x$ modeling the log-odds ratio in standard logistic regression is replaced by the Choquet integral. Formally, this leads to the following model:

$$\pi_c \stackrel{\text{df}}{=} \mathbf{P}(y = 1 \,|\, x) = \frac{1}{1 + \exp\Big(-\gamma\big(\mathscr{C}_\mu(f_x) - \beta\big)\Big)} \,, \tag{8}$$

where $\mathscr{C}_\mu(f_x)$ is the Choquet integral (with respect to the measure μ) of the evaluation function $f_x : \{c_1,\ldots,c_m\} \to [0,1]$ that maps each attribute c_i to a value $x_i = f_x(c_i)$; $\beta,\gamma \in \mathbb{R}_+$ are constants. The value of x_i is normalized in order to turn each predictor variable into a criterion, i.e., a "the higher the better" attribute, and to assure commensurability between the criteria [18].

The model (8) has several degrees of freedom, namely the fuzzy measure μ (Möbius transform $m = m_\mu$), the threshold β and the scaling parameter γ. The goal of learning is to identify these degrees of freedom on the basis of the training data $\mathscr{D}$. Like in the case of standard logistic regression, it is possible to harness the maximum likelihood (ML) principle for this purpose. The log-likelihood of the parameters can be written as

$$\begin{aligned} l(m,\gamma,\beta) &= \log \mathbf{P}(\mathscr{D} \,|\, m,\beta,\gamma) \\ &= \log\left(\prod_{i=1}^{n} \mathbf{P}(y^{(i)} \,|\, x^{(i)}; m,\beta,\gamma)\right) \\ &= \sum_{i=1}^{n} y^{(i)} \log \pi_c^{(i)} + \left(1-y^{(i)}\right) \log\left(1-\pi_c^{(i)}\right). \end{aligned} \tag{9}$$

This is a convex function with respect to m,γ, and β. The problem, now, is to maximize (9) while making sure that μ is a proper fuzzy measure. Formally, this leads to the following constrained optimization problem:

$$\max_{m,\gamma,\beta} \left\{ -\gamma \sum_{i=1}^{n} (1-y^{(i)}) \left(\mathscr{C}_m(x^{(i)}) - \beta\right) - \sum_{i=1}^{n} \log\left(1+\exp(-\gamma(\mathscr{C}_m(x^{(i)}) - \beta))\right) \right\}$$

such that

$$0 \le \beta \le 1$$

$$0 < \gamma$$

$$\sum_{T \subseteq C} m(T) = 1 \tag{10}$$

$$\sum_{B \subseteq A \setminus \{c_i\}} m(B \cup \{c_i\}) \ge 0 \quad \forall A \subseteq C, c_i \in A \tag{11}$$

4 Efficient Learning of 2-Additive Measures

Solving the above optimization problem is a non-trivial task and may become computationally expensive, mainly due to the constraints on the fuzzy measure μ.

In fact, since (11) needs to be satisfied for all subsets $A \subseteq C$, the number of these monotonicity constraints is given by $m2^{m-1}$ and thus grows exponentially with the number of attributes.

In the following, we restrict ourselves to the specific case of 2-additive fuzzy measures. This restriction is interesting for several reasons. In particular, one may of course hope for a gain in terms of computational efficiency, and indeed, this is the aspect that we shall focus on in the remainder of the paper. Besides, however, let us mention that a restriction of this kind is also interesting from a learning point of view: By allowing one to capture pairwise interactions between attributes, the 2-additive case is a proper generalization of the linear model, while at the same time, it is still reasonable in terms of the number of degrees of freedom. In fact, while the number of parameters to be estimated is exponential (in the number of attributes) in general, it is only quadratic in the 2-additive case. Practically, we could observe that the high flexibility of the general model is rarely needed; on the contrary, it often leads to problems of over-fitting the data, thereby compromising generalization performance.

Coming back to the computational aspect, the number of parameters to be estimated is indeed reduced, since $m(A) = 0$ for all $A \subseteq C$ such that $|A| > 2$. On the other hand, it is important to observe that the number of constraints does *not* reduce: Although the number of summands in each of the constraints (11) becomes smaller (since many of them are now 0), the number of constraints themselves remains the same.

In the following, we shall therefore look for ways to exploit the simplified structure of the 2-additive case in order to reduce the number of constraints. More specifically, we shall propose two alternative formulations of the constraint optimization problem to be solved for ML estimation.

4.1 Alternative Formulation I

To simplify notation, let $C = \{1,\ldots,m\}$ (instead of $C = \{c_1,\ldots,c_m\}$) and let $\mathscr{M}$ denote the class of nonnegative monotone set functions on C, i.e., the class of functions $\nu : 2^C \to [0,\infty)$ such that $\nu(A) \leq \nu(B)$ for all $A \subseteq B \subseteq C$; for the time being, we neglect the normalization condition (10), as it is less important for our purpose (it constitutes a single constraint that must be added to the optimization problem in order to turn a monotone measure into a fuzzy measure). More specifically, we are interested in the subclass $\mathscr{M}_2 \subset \mathscr{M}$ of 2-additive measures ν, i.e., whose Möbius transform satisfies $m_\nu(A) = 0$ for all $A \subseteq C$ such that $|A| > 2$.

The following characterization is well-known (see, e.g., Proposition 1 in [16]): $\nu \in \mathscr{M}_2$ if and only if the following constraints $C_{i,X}$ are satisfied for all $i \in C$ and $X \subseteq C_i = C \setminus \{i\}$:

$$C_{i,X}: \quad m_i + \sum_{j \in X} m_{i,j} \geq 0 \ , \tag{12}$$

where $m_i = m_v(\{i\})$ and $m_{i,j} = m_v(\{i,j\})$. Note that the number of constraints (12) is still exponential in m. Yet, we can show that they can be expressed equivalently in terms of a smaller number of constraints (albeit at the expense of introducing additional variables).

Proposition 1. *Condition (12) is equivalent to the following condition: For all $i \in C$, there exist $\alpha_{i,j}$, $j \in C_i$, such that*

$$\begin{aligned} &\alpha_{i,j} \geq 0 \\ &\sum_{j \in C_i} \alpha_{i,j} \leq 1 \\ &m_i \geq 0 \\ &m_{i,j} \geq -\alpha_{i,j} \cdot m_i \end{aligned} \tag{13}$$

Proof: Let $v \in \mathscr{M}_2$ and suppose (12) to hold. For $i \in C$, (12) with $X = \emptyset$ implies $m_i \geq 0$. Now, define $C_i^- = \{j \in C_i \,|\, m_{i,j} < 0\}$, $C_i^+ = \{j \in C_i \,|\, m_{i,j} \geq 0\}$, and let

$$\alpha_{i,j} = \begin{cases} 0 & \text{if } j \in C_i^+ \\ \frac{|m_{i,j}|}{m_i} & \text{if } j \in C_i^- \end{cases}$$

Since (12) holds with $X = C_i^-$, we have

$$\sum_{j \in C_i^-} |m_{i,j}| \leq m_i \ ,$$

and therefore

$$\sum_{j \in C_i} \alpha_{i,j} = \sum_{j \in C_i^-} \alpha_{i,j} = \sum_{j \in C_i^-} \frac{|m_{i,j}|}{m_i} = \frac{1}{m_i} \sum_{j \in C_i^-} |m_{i,j}| \leq 1.$$

Moreover, $m_{i,j} \geq -\alpha_{i,j} \cdot m_i$ holds by definition, both for $j \in C_i^+$ and $j \in C_i^-$. Thus, condition (13) holds, and hence (12) implies (13).

Now, suppose that (13) holds. Then, $m_i \geq 0$ and for any $\emptyset \neq X \subseteq C_i$,

$$\begin{aligned} m_i + \sum_{j \in X} m_{i,j} &\geq m_i + \sum_{j \in X} -\alpha_{i,j} \cdot m_i \\ &= m_i - m_i \sum_{j \in X} \alpha_{i,j} \\ &\geq m_i (1 - \sum_{j \in X} \alpha_{i,j}) \geq 0 \end{aligned}$$

Thus, condition (12) holds, and hence (13) implies (12). Q.E.D.

As a consequence of the above result, the constraints (11) can be replaced by the equivalent constraints (13). Thus, the number of constraints can indeed be reduced from exponential to quadratic, namely to $2m^2$ inequalities. On the other hand,

(13) also comes with a disadvantage: While the constraints (11) are all linear, some of the constraints (13) are *nonlinear* (albeit convex); indeed, recall that the $\alpha_{i,j}$ are introduced as new variables that need to be determined simultaneously with the m_i and $m_{i,j}$.

4.2 Alternative Formulation II

Our second reformulation of the problem is based on a theoretical result showing that the class $\mathscr{M}_2$ or, more specifically, the class of normalized measures in $\mathscr{M}_2$ (i.e., those ν whose Möbius function additionally satisfies (10), forms a convex polytope. The extreme points of this polytope are exactly those $\{0,1\}$-valued measures whose Möbius transforms are of the form

$$m_A(X) = \begin{cases} 1 & \text{if } X = A \\ 0 & \text{otherwise} \end{cases}, \qquad A \in \mathscr{E}$$

or of the form

$$m'_B(X) = \begin{cases} 1 & \text{if } \emptyset \neq X \subsetneq B \\ -1 & \text{if } X = B \\ 0 & \text{otherwise} \end{cases}, \qquad A \in \mathscr{E}',$$

where $\mathscr{E} = \{A \subseteq C \,|\, 1 \leq |A| \leq 2\}$ and $\mathscr{E}' = \{B \subseteq C \,||B| = 2\}$ [17]. In other words, each feasible solution m can be written as a convex combination of these m^2 extreme points:

$$m = \sum_{A \in \mathscr{E}} \alpha_A \cdot m_A + \sum_{B \in \mathscr{E}'} \alpha'_B \cdot m'_B \tag{14}$$

Consequently, the constraints (10)–(11) can be replaced by (14) in conjunction with the following constraints:

$$\begin{aligned} &\alpha_A \geq 0 \\ &\alpha'_B \geq 0 \\ &\sum_{A \in \mathscr{E}} \alpha_A + \sum_{B \in \mathscr{E}'} \alpha_B = 1 \end{aligned}$$

Like in our first reformulation, the number of constraints is thus significantly reduced, this time even without introducing nonlinearities, albeit again at the cost of a quadratic number of additional variables. More concretely, we end up with m^2 additional variables while reducing the number of constraints to $m^2 + 1$.

5 Experiments

The collection of data for experimental evaluation is a bit hindered by the fact that choquistic regression is a method for learning *monotone models*, i.e., models in which the probability of a positive output is an increasing function of each

Table 1 Data sets and their properties

	data set	#instances	#attributes	source
1	Employee Selection (ESL)	488	4	WEKA
2	Employee Rejection/Acceptance (ERA)	1000	4	WEKA
3	Lecturers Evaluation (LEV)	1000	4	WEKA
4	CPU	209	6	UCI
5	Mammographic (MMG)	961	5	UCI
6	Car Evaluation (CEV)	1728	6	UCI
7	Auto MPG	392	7	UCI
8	Den Bosch (DBS)	120	8	[5]
9	Breast Cancer (BCC)	286	7	UCI
10	Social Workers Decisions (SWD)	1000	10	[6]

Table 2 Classification accuracy in terms of 0/1 loss (mean $\pm$ standard deviation derived from 10 repeats of 5-fold cross-validation)

data set	CR-orig	CR-AI	CR-AII	LR
ESL	$.0655 \pm .0225$	$.0668 \pm .0227$	$.0639 \pm .0208$	$.0678 \pm .0255$
ERA	$.2908 \pm .0312$	$.2880 \pm .0292$	$.2907 \pm .0312$	$.2873 \pm .0275$
LEV	$.1478 \pm .0202$	$.1491 \pm .0222$	$.1530 \pm .0213$	$.1686 \pm .0240$
CPU	$.0241 \pm .0223$	$.0244 \pm .0197$	$.0196 \pm .0236$	$.0672 \pm .0346$
MMG	$.1685 \pm .0240$	$.1697 \pm .0232$	$.1661 \pm .0232$	$.1712 \pm .0268$
CEV	$.0743 \pm .0127$	$.0835 \pm .0120$	$.0726 \pm .0135$	$.1382 \pm .0170$
MPG	$.0663 \pm .0244$	$.0644 \pm .0281$	$.0636 \pm .0254$	$.0627 \pm .0277$
DBS	$.1413 \pm .0715$	$.1330 \pm .0648$	$.1130 \pm .0645$	$.1472 \pm .0573$
BCC	$.3041 \pm .0581$	$.2840 \pm .0556$	$.3065 \pm .0524$	$.3079 \pm .0586$
SWD	$.2186 \pm .0187$	$.2169 \pm .0276$	$.2143 \pm .0225$	$.2202 \pm .0244$

input attribute. Data sets for which monotonicity of this kind is a reasonable assumption are less frequent than standard classification data. Nevertheless, we managed to collect 10 such data sets; Table 1 provides a summary of their main properties. Those with a numerical or ordered categorical output were binarized by thresholding at the median. Moreover, all input attributes were normalized.

Experimentally, we compared three versions of choquistic regression, the original formulation from Section 3.1 (CR-orig), the first reformulation from Section 4.1 (CR-AI), and the second reformulation from Section 4.2 (CR-AII). To make the implementations as comparable as possible, we applied the same solver to the different optimization problems, namely the `fmincon` function implemented in the optimization toolbox of Matlab. This function provides a method for constrained nonlinear optimization based on sequential quadratic programming.

In terms of classification accuracy, the different implementations of choquistic regression should perform exactly the same, at least theoretically, because they seek to maximize the same likelihood function under different but equivalent constraints.

Table 3 Runtime complexity of the alternative implementations on different data sets (name, number of attributes, number of instances) measured in terms of CPU time (mean ± standard deviation in seconds) for different sample sizes (in % of the complete data set)

data	CR	20%	40%	60%	80%	100%
ESL	orig	0.26 ± 0.05	0.31 ± 0.02	0.38 ± 0.02	0.45 ± 0.13	0.63 ± 0.05
4	AI	0.41 ± 0.13	0.50 ± 0.07	0.68 ± 0.13	0.80 ± 0.17	1.05 ± 0.18
488	AII	0.31 ± 0.09	0.39 ± 0.07	0.50 ± 0.06	0.61 ± 0.04	0.70 ± 0.04
ERA	orig	0.23 ± 0.03	0.36 ± 0.01	0.50 ± 0.02	0.63 ± 0.01	0.78 ± 0.02
4	AI	0.53 ± 0.10	0.90 ± 0.08	1.06 ± 0.16	1.20 ± 0.20	1.35 ± 0.18
1000	AII	0.31 ± 0.05	0.52 ± 0.07	0.70 ± 0.09	1.12 ± 0.14	1.32 ± 0.16
LEV	orig	0.34 ± 0.04	0.55 ± 0.05	0.71 ± 0.04	0.88 ± 0.07	1.03 ± 0.07
4	AI	0.96 ± 0.23	1.41 ± 0.21	1.84 ± 0.24	2.25 ± 0.18	2.50 ± 0.19
1000	AII	0.49 ± 0.07	0.76 ± 0.05	1.04 ± 0.10	1.68 ± 0.15	1.90 ± 0.14
CPU	orig	0.77 ± 0.18	1.95 ± 3.39	3.37 ± 5.42	6.9 ± 8.97	14.23 ± 11.33
6	AI	1.85 ± 0.22	2.56 ± 0.52	2.79 ± 0.71	3.42 ± 0.18	6.11 ± 2.71
209	AII	0.50 ± 0.31	1.28 ± 0.24	1.33 ± 0.29	1.68 ± 0.56	2.06 ± 0.66
MMG	orig	0.39 ± 0.15	0.56 ± 0.06	0.79 ± 0.12	0.95 ± 0.09	1.07 ± 0.11
6	AI	1.19 ± 0.24	1.77 ± 0.47	2.06 ± 0.61	2.71 ± 1.60	3.24 ± 1.96
961	AII	0.52 ± 0.13	0.83 ± 0.11	1.13 ± 0.10	1.54 ± 0.18	1.78 ± 0.19
CEV	orig	2.45 ± 0.24	3.84 ± 0.38	5.09 ± 0.41	5.79 ± 0.51	6.74 ± 0.41
6	AI	5.36 ± 0.55	7.53 ± 1.00	9.89 ± 0.96	11.93 ± 2.83	13.72 ± 2.56
1728	AII	2.11 ± 0.33	3.68 ± 0.31	5.23 ± 0.52	6.88 ± 0.59	7.88 ± 0.58
MPG	orig	1.83 ± 0.71	2.15 ± 0.62	2.69 ± 0.59	3.18 ± 0.54	3.45 ± 0.65
7	AI	2.58 ± 0.32	2.54 ± 0.66	3.46 ± 0.89	3.84 ± 0.75	4.15 ± 0.92
392	AII	0.61 ± 0.21	0.72 ± 0.12	0.95 ± 0.24	1.02 ± 0.19	1.3 ± 0.13
DBS	orig	5.68 ± 1.11	5.36 ± 1.23	5.61 ± 1.02	5.59 ± 0.72	5.47 ± 1.05
8	AI	2.51 ± 1.81	2.88 ± 1.29	3.03 ± 1.42	3.17 ± 0.96	4.08 ± 1.10
120	AII	0.71 ± 0.19	0.78 ± 0.34	0.76 ± 0.18	0.82 ± 0.12	0.91 ± 0.13
BCC	orig	1.22 ± 0.56	1.10 ± 0.27	1.19 ± 0.23	1.47 ± 0.38	1.47 ± 0.25
9	AI	2.29 ± 1.09	2.04 ± 1.52	2.16 ± 0.95	2.88 ± 2.5	2.97 ± 2.30
286	AII	0.47 ± 0.24	0.47 ± 0.06	0.55 ± 0.55	0.66 ± 0.11	0.78 ± 0.07
SWD	orig	292.4 ± 31.1	382.8 ± 42.24	371.3 ± 12.67	394.0 ± 36.62	427.5 ± 36.62
10	AI	17.9 ± 13.4	27.82 ± 12.13	32.11 ± 10.10	32.35 ± 10.05	33.14 ± 10.77
1000	AII	4.7 ± 0.71	8.80 ± 1.34	13.01 ± 1.44	18.24 ± 2.21	22.66 ± 1.73

Practically, of course, different formulations of the optimization problem will yield slightly different solutions, although these differences should be small. This expectation is confirmed by the result of a 5-fold cross validation, which is summarized in Table 2; this table also shows results for standard logistic regression (LR) as a baseline.

What we are of course most interested in is the runtime performance of the different implementations, which we measured in terms of CPU usage.[1] The results, which are summarized in Table 3, convey a quite clear picture: While the original implementation CR-orig is superior or at least competitive for data sets with up to 6 attributes, it is visibly outperformed by the alternative formulations for $m > 6$ attributes, and the difference in runtime rapidly increases with m. This is in agreement with our expectations: An exponential number of constraints is no big obstacle provided the number of attributes is small. In this case, a reduction from exponential to quadratic does not compensate for the additional overhead caused by introducing new variables. Due to the exponential growth of the number of constraints in CR-orig, however, this situation quickly changes in favor of CR-AI and CR-AII with an increasing number of attributes; indeed, as can be seen from the SWD data, the runtime of CR-orig becomes unacceptable as soon as $m > 9$.

This is also confirmed by another experiment we did with this data set: From the total of 10 attributes, we randomly samples $m \in \{5,6,\ldots,10\}$, trained a CR model on the data set reduced to these k attributes (using the tree methods CR-orig, CR-AI and CR-AII) and measured the runtime. This was repeated many times and the runtime was averaged. Fig. 1 shows this average runtime as a function of m.

Comparing the two alternatives CR-AI and CR-AII, it seems that the latter is consistently faster, although the growth of the runtime as a function of m is in both cases much more moderate than for CR-orig. Again, this is not unexpected against the background of the results from the previous section.

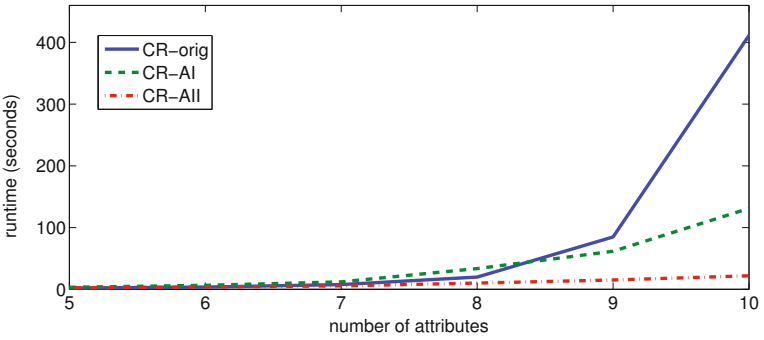


Fig. 1 Average runtime on the SWD data as a function of the number of attributes included

6 Discussion

Our experimental results are in complete agreement with the theoretical complexity (in terms of the number of constraints and the number of variables involved) of the optimization problems. Thus, learning the Choquet integral for classification can indeed be made more efficient by exploiting the special structure of the problem

[1] Experiments were carried out on an Intel Core(TM) i7-2600 CPU with 3.40GHz and 8 GB RAM under Windows 7.

in the case of 2-additive fuzzy measures, essentially reducing the complexity from exponential to quadratic in the number of attributes.

In order to compare the different variants of the problem (CR-orig, CR-AI, CR-AII), we decided to use a rather general optimization method that can handle all of them without the need for specific adaptations. An interesting alternative, of course, is to implement each of the variants individually and as efficiently as possible, seeking for a more specialized solver that allows for exploiting the respective problem structure in an optimal way. In particular, this appears to be important for a more thorough comparison of the two alternatives we proposed, respectively, in Sections 4.1 and 4.2.

Theoretically, CR-AII seems to be advantageous to CR-AI, and indeed, the experimental results are in agreement with this presumption. Nevertheless, the reformulation in Section 4.1 should not be abandoned rashly. First, as just mentioned, it might be possible to improve its efficiency by means of specialized optimization techniques; one may think, for example, of an alternating optimization scheme in which, repeatedly, the $\alpha_{i,j}$ are fixed while the $m_{i,j}$ are optimized and vice versa, thereby circumventing the issue of nonlinearity.

Moreover, CR-AII might be more amenable for a generalization to the case of k-additive measures, $k > 2$. In this regard, the second approach is arguably difficult: Firstly, it is known that for $k > 2$, the extreme points of the convex polytope of k-additive measures are not all $\{0,1\}$-valued. Secondly, and more importantly, the number of these extreme points is expected to grow extremely fast, knowing that the number of extreme points of the polytope of additive measures on m variables grows like the sequence of Dedekind numbers [15].

Acknowledgements. This work was supported by the German Research Foundation (DFG).

References

[1] Angilella, S., Greco, S., Matarazzo, B.: Non-additive robust ordinal regression with Choquet integral, bipolar and level dependent Choquet integrals. In: Proc. IFSA/EUSFLAT 2009, Lisbon, Portugal, pp. 1194–1199 (2009)

[2] Beliakov, G., James, S.: Citation-based journal ranks: the use of fuzzy measures. Fuzzy Sets and Systems 167(1), 101–119 (2011)

[3] Ben-David, A.: Monotonicity maintenance in information-theoretic machine learning algorithms. Machine Learning 19, 29–43 (1995)

[4] Choquet, G.: Theory of capacities. Annales de l'institut Fourier 5, 131–295 (1954)

[5] Daniels, H., Kamp, B.: Applications of MLP networks to bond rating and house pricing. Neural Computation and Applications 8, 226–234 (1999)

[6] David, A.B.: (2010), http://mldata.org/repository/data/viewslug/datasets-arie_ben_david-swd/ (last accessed on June 21, 2012)

[7] Tehrani, A.F., Cheng, W., Dembczy, K., Hüllermeier, E.: Learning Monotone Nonlinear Models Using the Choquet Integral. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011. LNCS, vol. 6913, pp. 414–429. Springer, Heidelberg (2011)

[8] Fallah Tehrani, A., Cheng, W., Hüllermeier, E.: Choquistic regression: Generalizing logistic regression using the Choquet integral. In: 7th Int. Conf. of the European Society for Fuzzy Logic and Technology, EUSFLAT 2011, Aix-les-Bains, France, pp. 868–875 (2011)
[9] Fallah Tehrani, A., Cheng, W., Hüllermeier, E.: Preference learning using the Choquet integral: The case of multipartite ranking. IEEE Transactions on Fuzzy Systems (forthcoming, 2012)
[10] Feelders, A.: Monotone relabeling in ordinal classification. In: Proc. of the 10th IEEE International Conference on Data Mining, pp. 803–808. IEEE Press, Piscataway (2010)
[11] Grabisch, M.: Fuzzy integral in multicriteria decision making. Fuzzy Sets and Systems 69(3), 279–298 (1995)
[12] Grabisch, M.: Modelling data by the Choquet integral. In: Torra, V. (ed.) Information Fusion in Data Mining, pp. 135–148. Springer, Heidelberg (2003)
[13] Grabisch, M., Nicolas, J.M.: Classification by fuzzy integral: performance and tests. Fuzzy Sets and Systems 65(2-3), 255–271 (1994)
[14] Grabisch, M., Murofushi, T., Sugeno, M.: Fuzzy Measures and Integrals: Theory and Applications. Physica-Verlag, Heidelberg (2000)
[15] Miranda, P., Grabisch, M.: On vertices of the k-additive monotone core. In: Proc. IFSA/EUSFLAT 2009, Lisbon, Portugal, pp. 76–81 (2009)
[16] Miranda, P., Grabisch, M., Gil, P.: Axiomatic structure of k-additive capacities. Mathematical Social Sciences 49, 153–178 (2005)
[17] Miranda, P., Combarro, E.F., Gil, P.: Extreme points of some families of non-additive measures. European J. of Operational Research 174, 1865–1884 (2006)
[18] Modave, F., Grabisch, M.: Preference representation by a Choquet integral: commensurability hypothesis. In: Proc. of the 7th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, pp. 164–171 (1998)
[19] Potharst, R., Feelders, A.: Classification trees for problems with monotonicity constraints. ACM SIGKDD Explorations Newsletter 4(1), 1–10 (2002)
[20] Sugeno, M.: Theory of fuzzy integrals and its application. PhD thesis, Tokyo Institute of Technology, Tokyo, Japan (1974)
[21] Torra, V.: Learning aggregation operators for preference modeling. In: Fürnkranz, J., Hüllermeier, E. (eds.) Preference Learning, pp. 317–333. Springer, Heidelberg (2011)
[22] Torra, V., Narukawa, Y.: Modeling Decisions: Information Fusion and Aggregation Operators. Springer, Heidelberg (2007)
[23] Vitali, G.: Sulla definizione di integrale delle funzioni di una variabile. Annali di Matematica Pura ed Applicata 2(1), 111–121 (1925)

Can Fuzzy Clustering Avoid Local Minima and Undesired Partitions?

Balasubramaniam Jayaram and Frank Klawonn

Abstract. Empirical evaluations and experience seem to provide evidence that fuzzy clustering is less sensitive w.r.t. to the initialisation than crisp clustering, i.e. fuzzy clustering often tends to converge to the same clustering result independent of the initialisation whereas the result for crisp clustering is highly dependent on the initialisation. This leads to the conjecture that the objective function used for fuzzy clustering has less undesired local minima than the one for hard clustering. In this paper, we demonstrate that fuzzy clustering does suffer from unwanted local minima based on concrete examples and show how these undesired local minima of the objective function in fuzzy clustering can vanish by using a suitable value for the fuzzifier.

1 Introduction

The aim of cluster analysis is to construct a partition of a given data set into homogenous groups, called clusters. Data objects within a cluster should be similar, whereas data objects assigned to different clusters should differ significantly. The main motivation for the introduction of fuzzy clustering as a generalisation of crisp

Balasubramaniam Jayaram
Department of Mathematics, Indian Institute of Technology Hyderabad,
Yeddumailaram 502205, India
e-mail: jbala@iith.ac.in

Frank Klawonn
Department of Computer Science, Ostfalia University of Applied Sciences,
38302 Wolfenbuettel, Germany
e-mail: f.klawonn@ostfalia.de
and
Bioinformatics and Statistics, Helmholtz Centre for Infection Research,
38124 Braunschweig, Germany
e-mail: frank.klawonn@helmholtz-hzi.de

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 31–44.
springerlink.com

or partitioning clustering was to better represent partly overlapping clusters. Data points at the boundary between two clusters should belong partly to both clusters.

Apart from this obvious motivation for fuzzy clustering, it seems that fuzzy clustering is more robust in the sense that the results seem to be less dependent on the initialisation that is required for many clustering algorithms. Since fuzzy clustering is usually based on minimising an objective function by a gradient descent method, this empirical observation suggests the conclusion that the fuzzy versions of crisp clustering algorithms have less local minima in which the clustering algorithm can get stuck.

First investigations in this direction have been described in [14], but without final proofs that local minima of the objective function can really vanish in fuzzy clustering. After a brief review of fuzzy cluster analysis, we provide concrete examples where it can be clearly observed that undesired local minima of the objective function can be ruled out by fuzzy clustering. Although this is a positive result, new problems are introduced by fuzzy clustering when applied to high-dimensional data.

2 From Crisp to Fuzzy Clustering

A simple and common popular approach is the so-called c-means clustering (HCM)[1] [8]. For the HCM algorithm it is assumed that the number of clusters is known or at least fixed, i.e., the algorithm will partition a given data set $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ into c clusters. Since the assumption of a known or a priori fixed number of clusters is not realistic for many data analysis problems, there are techniques based on cluster validity considerations that allow to determine the number of clusters for the HCM algorithm as well. A comparison of methods for determining the number of clusters can be found in [6]. In recent years, resampling or cross-validation techniques [5] are often used to determine the number of clusters. However, the underlying algorithm remains more or less the same, only the number of clusters is varied and the resulting clusters or the overall partition is evaluated. Therefore, it is sufficient to assume for the rest of the paper that the number of clusters is always fixed.

From the purely algorithmic point of view, the c-means clustering can be described as follows. Each of the c clusters is represented by a prototype $v_i \in \mathbb{R}^m$. These prototypes are chosen randomly in the beginning. Then each data vector is assigned to the nearest prototype (w.r.t. the Euclidean distance). Then each prototype is replaced by the centre of gravity of those data assigned to it. The alternating assignment of data to the nearest prototype and the update of the prototypes as cluster centres is repeated until the algorithm converges, i.e., no more changes happen.

This algorithm can also be seen as a strategy for minimising the following objective function:

$$f = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij} d_{ij} \tag{1}$$

[1] Usually, the algorithm is called k-means. But in fuzzy clustering it is common to use the letter c instead of k for the number of clusters. HCM stand for *H*ard *C*-*M*eans clustering in order to distinguish it from *F*uzzy *C*-*M*eans clustering (FCM).

under the constraints

$$\sum_{i=1}^{c} u_{ij} = 1 \qquad \text{for all } j = 1, \ldots, n \tag{2}$$

where $u_{ij} \in \{0,1\}$ indicates whether data vector x_j is assigned to cluster i ($u_{ij} = 1$) or not ($u_{ij} = 0$). $d_{ij} = \| x_j - v_i \|^2$ is the squared Euclidean distance between data vector x_j and cluster prototype v_i.

It would be a straight forward generalisation of HCM to simply relax the constraints $u_{ij} \in \{0,1\}$ to $u_{ij} \in [0,1]$ in order to obtain a fuzzy version of HCM. However, it turned out that the minimum of the objective function (1) under the constraints (2) is still obtained, when u_{ij} is chosen in the same way as in HCM, i.e. $u_{ij} \in \{0,1\}$, even if we allow $u_{ij} \in [0,1]$. Therefore, an additional parameter w, the so-called fuzzifier, was introduced – first only for the choice $w = 2$ [9] and later on for any $w > 1$ [2] – and the objective function (1) is replaced by

$$f = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} d_{ij}. \tag{3}$$

Note that the fuzzifier w does not have any effects, when we use hard clustering. The fuzzifier $w > 1$ is not subject of the optimisation process and has to be chosen in advance. A typical choice is $w = 2$.

The minimisation of the objective function (3) under the constraints (2) is usually carried out by an alternating optimisation scheme where the membership degrees are updated by

$$u_{ij} = \left(\frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}} \right)^{\frac{1}{w-1}}} \right)^{w}, \tag{4}$$

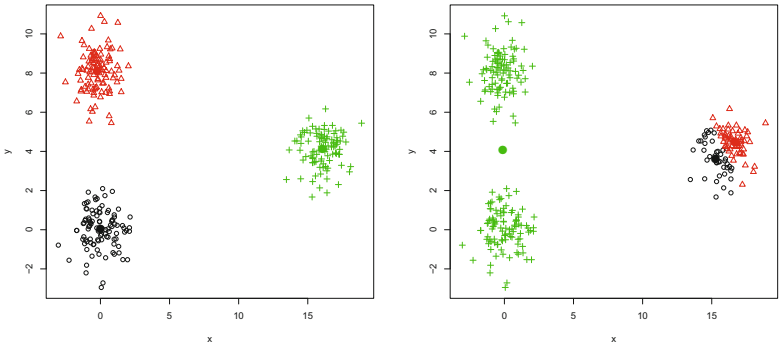
and – in case of the Euclidean distance – the cluster prototypes by

$$v_i = \frac{\sum_{j=1}^{n} u_{ij}^{w} x_j}{\sum_{j=1}^{n} u_{ij}^{w}}. \tag{5}$$

This is the standard fuzzy c-means algorithm (FCM). The update equations (4) and (5) represent the global minimum of the objective function when the corresponding other set of parameters is considered as fixed.

Fig. 1 shows a simple data set with three well-separated clusters. However, in 2,589 out of 10,000 runs with random initialisation, HCM gets stuck in a local minimum of the objective function leading to the undesired clustering result shown in Fig. 1(b) whereas FCM terminates in the correct partition (a) in all 10,000 runs[2]. The reason for the failure of HCM lies in the fact that once a prototype has

[2] The clustering was carried out with the package `cluster` of the statistics software R [19].



(a) HCM works as expected (b) HCM fails!

Fig. 1 A simple two-dimensional data set and an HCM clustering result as it is expected (a). But in about 25% of the runs, HCM gets stuck in a local mininum leading to the partition (b).

'conquered' the two clusters in left-hand side, the other two prototypes will not take any notice of these points anymore.

It is out of the scope of this paper to provide a detailed review on fuzzy clustering as for instance in [3, 12]. It should be noted that there are two parts of the objective function (3) that can be modified or generalised. One the one hand, there is the way how fuzzy membership degrees are incorporated in the objective function. Possibilistic clustering [17] relaxes the constraints (2), leading to an actually undesired global minimum. An improved version of possibilistic clustering, avoiding the problem, has been proposed in [20] for the price of significantly higher computational costs. In [16, 15], the fuzzifier is replaced by more general functions than just a simple power of the membership degrees to overcome certain problems that are introduced by the fuzzifier. One of these problems is discussed at the end of Section 3.

On the other hand, the distance measure can be modified to cover more general cluster shapes. Various approaches have been proposed, for instance to adapt to linear [4, 2] or ellipsoidal [10] clusters, to clusters of different volume [13] or to non-compact shell clusters [18]. Although all these approaches have been published as fuzzy clustering techniques, they have actually nothing specific to do with fuzzy clustering. In principle, one could also use crisp membership degrees for them. The reason why these approaches are exclusively based on fuzzy clustering is probably that the more complex cluster shapes with additional parameters introduce more local minima into the objective function, so that there is a much higher risk to get stuck in an undesired local minimum when hard clustering is applied.

Noise clustering [7] is another example of an approach that is also applicable in the context of hard clustering. An additional noise cluster is introduced to which all data have a fixed (large) distance. In this way, data points that are far away from all clusters will be assigned to the noise cluster and have no longer any influence on other clusters.

3 Vanishing of Local Minima

As shown above HCM can get stuck in local minima if the initialisation is not 'proper'. While FCM certainly overcomes many of the lacunae in HCM, a similar problem can also plague FCM. For instance, is it true that FCM does not have any local minima? If it does, what is it that makes FCM come out of this? In this section, we firstly demonstrate that FCM does have undesired local minima and then argue that a proper fuzzifier can reduce the number of local minima in the objective function of FCM and thus help in the proper and faster convergence of FCM.

3.1 Local Minima of FCM

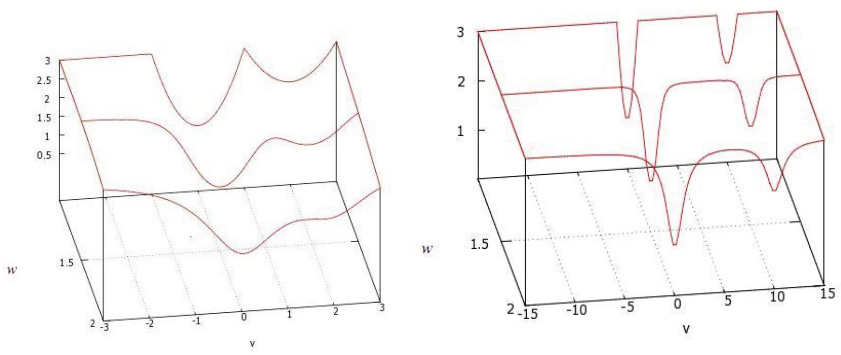
The objective function (3) of FCM is often difficult to visualise – there are too many dimensions (parameters, i.e., prototypes and membership degrees). Hence, let us reduce the dimensions by making the objective function independent of the membership degrees by choosing the optimal values for the membership degrees as in [11] by replacing u_{ij} in (3) by (4).

Taking a similar approach as in [14], let us consider a one-dimensional data set with one cluster at $x = 0$ with k points and one outlier at $x = u$. Clearly, we have just one cluster and let us add a noise cluster [7] to take care of the outlier. Now, the objective function in (3) becomes

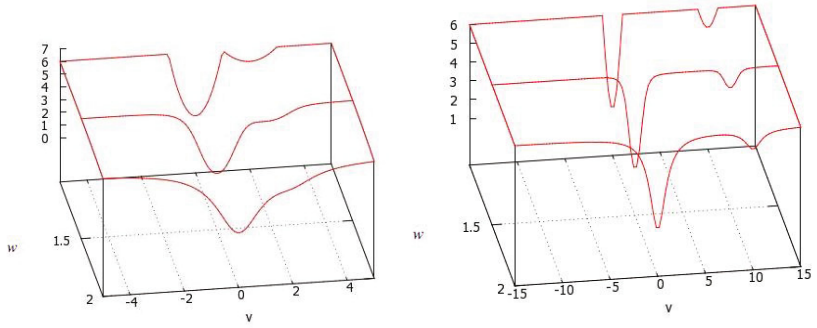
$$f(v) = \frac{k \cdot v^2}{\left(1 + \left(\frac{v^2}{\delta}\right)^{\frac{1}{w-1}}\right)^w} + \frac{(v-u)^2}{\left(1 + \left(\frac{(v-u)^2}{\delta}\right)^{\frac{1}{w-1}}\right)^w} + \frac{k \cdot \delta}{\left(1 + \left(\frac{\delta}{v^2}\right)^{\frac{1}{w-1}}\right)^w} + \frac{\delta}{\left(1 + \left(\frac{\delta}{(v-u)^2}\right)^{\frac{1}{w-1}}\right)^w} \tag{6}$$

where v is the location of the cluster centre and δ is the distance of every point to the noise cluster.

Let us consider Fig. 2, where the location of the cluster centre v is represented on the x-axis and the fuzzifier w on the y-axis. From Fig. 2(a), where $u = 2, k = 2$, i.e., the lone data point is at $x = 2$ and there are $k = 2$ points at $x = 0$, we see that when $w = 1$ there is a clear local minima at $v = 2$ while for the conventionally used value of $w = 2$ we see that the local minima is almost non-existent. Here the noise distance is $\delta = 1$. However, it does not mean that FCM is not plagued by this problem. To see this let us shift the lone data point from $v = 2$ to $v = 10$. As Fig. 2(b) shows, still with $\delta = 1$, we see a clear local minima at $v = 10$. Note that increasing the number of points at $x = 0$ does have an effect in the first case, as is expected, it does not have any effect in the second case, since the lone point is far enough not to be influenced by it. Moreover, note that the local density of data is not in our control in realistic



(a) $u = 2, k = 2$ (b) $u = 10, k = 2$



(c) $u = 2, k = 5$ (d) $u = 10, k = 5$

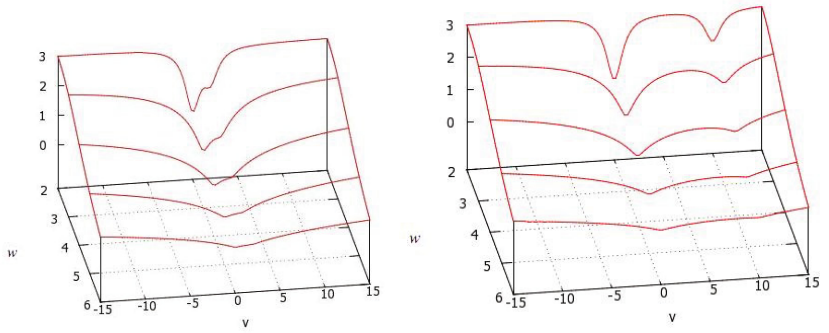
Fig. 2 Plots of the objective function of FCM for $w = 1, 1.5, 2$

situations and hence it is hard to ensure the vanishing of unwanted local minima. In fact, the distribution of the local density of data gives rise to an entirely different problem as is analysed and solved in [16, 14] (see below for more details).

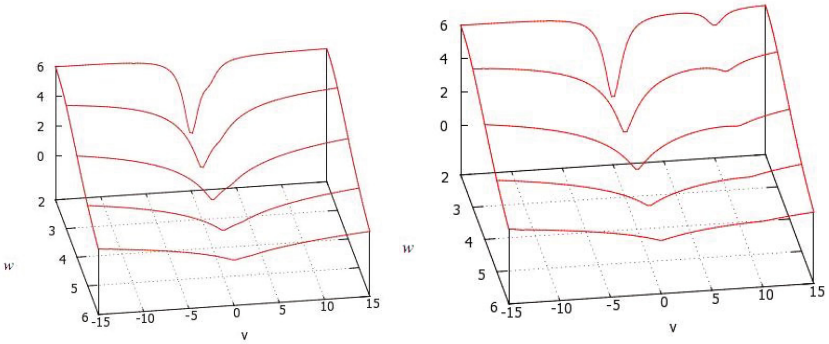
3.2 The Role of the Fuzzifier w

While the generalisation of the membership values u_{ij} from just $\{0, 1\}$ to the whole interval $[0, 1]$ is usually the highlighted aspect of FCM – perhaps even the nomenclature of FCM is also attributable to it – a major role is also played by the so called fuzzifier value 'w' in (3) above.

Firstly, note that even if $u_{ij} \in [0, 1]$ when $w = 1$ we still have hard clustering and FCM is equal to HCM. Secondly, as shown in [16] the value of the fuzzifier w actually controls the amount of overlap among the clusters. Looking at the term u_{ij}^w in (3) as only a particular transformation of u_{ij}, viz., $g(u) = u^w$, it was shown in [14] that suitable transformations g exist that also redeem FCM from the problem of letting their cluster centres be dictated by the local density of the data.



(a) $u = 2, k = 2$ (b) $u = 10, k = 2$



(c) $u = 2, k = 5$ (d) $u = 10, k = 5$

Fig. 3 Plots of the Objective Function of the FCM for $w = 2, 3, 4, 5, 6$

In this work, we show yet another aspect of the fuzzifier, viz., we show that choosing the w value appropriately can make many, if not all, of the local minima vanish and thus help FCM deliver correct results more often than not.

3.3 Suitable w for Vanishing Local Minima

Let us once again consider the scenarios presented in Sec. 3.1. As the Figs. 3(a)–(d) show, by increasing the value of w we see that the unwanted local minima at $v = 2$ and $v = 10$ vanish leading to a correct and, clearly, also a faster convergence. Note also that the local density of the data does not seem to alter the slope of the curve, equivalently the rate of convergence significantly. Thus it is very much applicable to real life data. While it can be seen from Fig. 3 that a value of around $w = 5$ or $w = 6$ seems sufficient to eliminate the unwanted local minima at $v = 2, 10$, it should be emphasised that the scenario considered here is very elementary. In higher dimensions, the value of w required could be much smaller or higher. For instance, see the scenario considerd in Section 3.4 below. As we understand the happenings while $w \to 1$, it is interesting to study the limiting case of $w \to \infty$.

To this end, it is sufficient to consider the following expression that occurs in the denominator of the objective function in (6): $\left(1+\left(\frac{d'}{d''}\right)^{\frac{1}{w-1}}\right)^{w}$, where d', d'' are positive distances of the points from the cluster centres. Since, when either $d'=0$ or $d''=0$ the expression does not come into play, we consider the following equivalent form: $g(w)=\left(1+(\varepsilon)^{\frac{1}{w-1}}\right)^{w}$ for $\varepsilon>0$. Now, it is obvious that $\lim_{w\to\infty} g(w)=\infty$ and hence $\lim_{w\to\infty} f(v)=0$ for every position v of the cluster centre. In other words, every point on the x-axis could be a cluster centre.

3.4 A Slightly More Complex Scenario

Let us consider the following scenario, where we have 2 clusters at $u=2$ and at $w=5$. There are 10 points each at $u=2$ and $w=5$ and 3 'noise' points at $x=0$. Using a noise cluster distance $\delta=1$, we expect the global minima to be at $\overline{v_1}=(2,5)$ and symmetrically at $\overline{v_2}=(5,2)$. Now the objective function becomes a two-variable function $F(\overline{v})=F(r,s)$, the formula of which is quite complex to be given here. However, we do plot F in Figs. 4 and 5 for different values of w. In every case there are clear global minima at $\overline{v_1}, \overline{v_2}$ as expected.

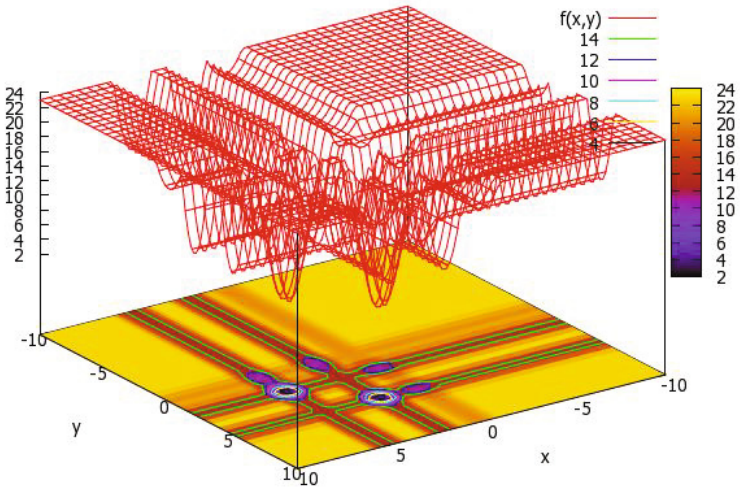
When $w=1$ (HCM), we see from Fig. 4(a) that, apart from the desired minima at $\overline{v_1}, \overline{v_2}$, there are also clear local minima around $(0,2),(2,0),(0,5),(5,0)$ as indicated by the dark blue contour circles. When $w=2$, as is usual for FCM, we see that two of the local minima have vanished and only the local minima around $\overline{v'}=(0,2)$ and $\overline{v''}=(2,0)$ remain. Thus if a cluster centre gets initialised closer to these local minima, it is difficult for these cluster centres to escape from there.

Now let us consider the case when $w=2.3$. It is already clear to see (see Fig. 5(a)) that most of the local minima have vanished and even if cluster centres fall close to the above $\overline{v'}, \overline{v''}$, they can eventually reach one of the global minima. This becomes even more apparent for the case when $w=2.9$ – see Fig. 5(b).

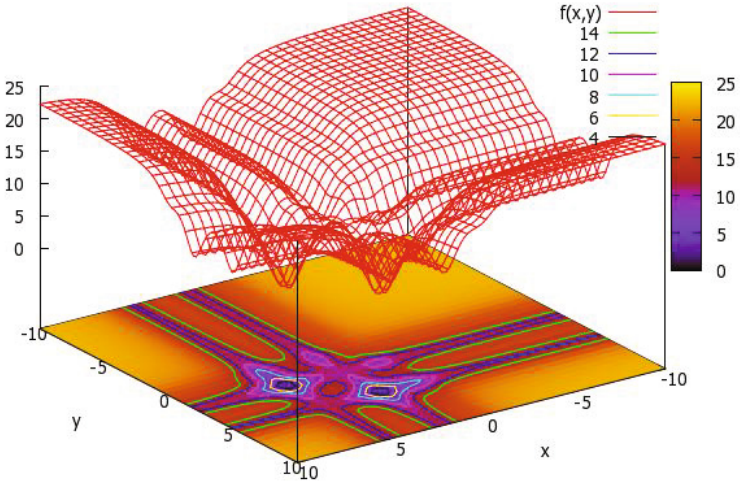
It is also interesting to note that for larger values of w, in fact, for $w=4$ or $w=5$ we see that all the local minima vanish and just one global minimum appears around the centroid of the whole data set, as is expected – see Figs. 6(a) & (b).

3.5 FCM Problems with High-Dimensional Data

The above examples seem to suggest that the fuzzifier will always lead to less local minima. However, for high-dimensional data, fuzzy clustering suffers from the so-called curse of dimensionality [1]. In higher dimensions, standard distances like the Euclidean distance seem to lose their power to distinguish between points. Fig. 7 is adopted from [21] where the following example is considered. Clusters are uniformly distributed on the surface of a hypersphere. Then the objective function of FCM is drawn along one axis only by moving the prototypes from the centre of the sphere along the radii to the cluster centres. Surprisingly, there is a local minimum of the objective function when all prototypes are positioned in the centre of

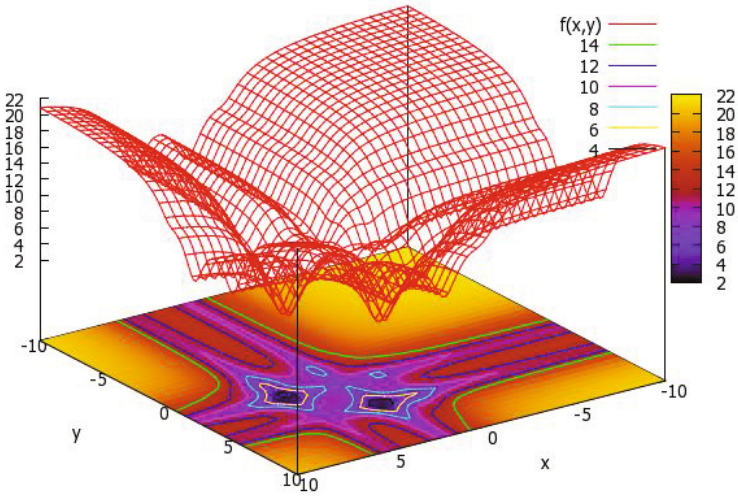


(a) $w = 1$, Local minima around $(0,2),(2,0),(0,5),(5,0)$

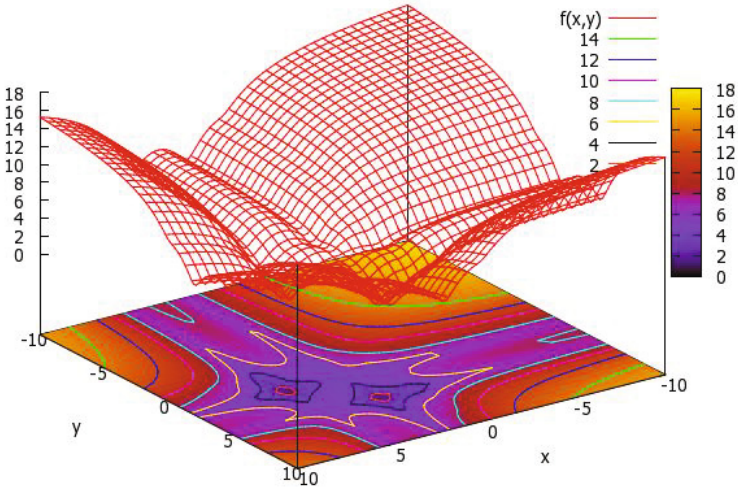


(b) $w = 2$, Local minima around $(0,2),(2,0)$

Fig. 4 Plots of the objective function $F(\overline{v})$ of FCM for $w = 1,2$

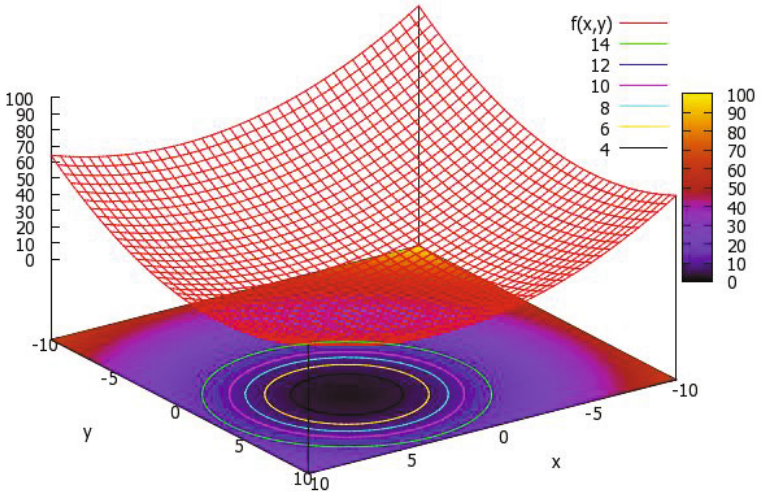


(a) $w = 2.3$

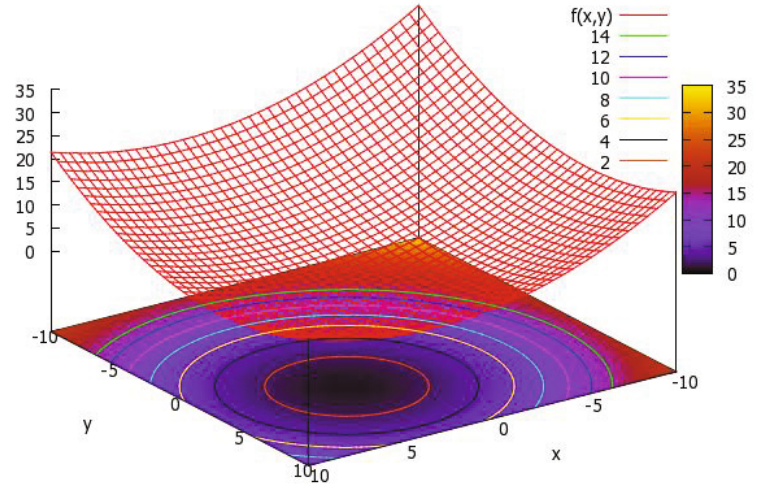


(b) $w = 2.9$, All the local minima have vanished

Fig. 5 Plots of the objective function $F(\overline{v})$ of FCM for $w = 2.3, 2.9$



(a) $w = 4$



(b) $w = 5$

Fig. 6 Plots of the objective function $F(\overline{v})$ of FCM for $w = 4, 5$ - one global minimum appears around the centroid of the whole data set

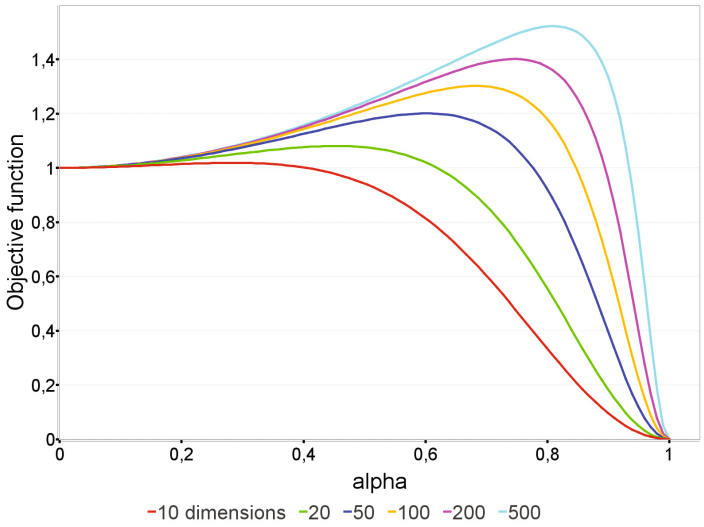


Fig. 7 An undesired local minimum at the centre 0 for high-dimensional data (the bottom curve is for dimension $m = 10$ and the top curve is for $m = 500$)

the sphere and the danger of getting stuck in this local minimum increases with number of dimensions. HCM does not have this specific problem although it has its own problems with high-dimensional data, see for instance [21] and the references therein.

One can escape from this problem of FCM with high-dimensional data by either choosing a fuzzifier very close to 1 or by using a generalised fuzzifier function as proposed in [16].

4 Conclusions

The answer to the initial question whether fuzzy clustering can avoid local minima is partly positive. For low-dimensional data, local minima can vanish by a suitable choice of the fuzzifier. For high-dimensional data, additional local minima can be introduced. The choice of the fuzzifier can be crucial for the avoidance of local minima. How to choose an appropriate value for the fuzzifier will be investigated in a future work.

Acknowledgements. This work was done during the visit of the first author to the Department of Computer Science, Ostfalia University of Applied Sciences under the fellowship provided by the Alexander von Humboldt Foundation. Part of this work was supported by DIeTY grant on Cyber Physical Systems–13(6)/2010-CC&BT.

References

[1] Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the Surprising Behavior of Distance Metrics in High Dimensional Space. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, pp. 420–434. Springer, Heidelberg (2000)

[2] Bezdek, J.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)

[3] Bezdek, J., Keller, J., Krishnapuram, R., Pal, N.: Fuzzy Models and Algorithms for Pattern Recognition and Image Processing. Kluwer, Boston (1999)

[4] Bock, H.: Clusteranalyse mit unscharfen partitionen. In: Bock, H. (ed.) Klassifikation und Erkenntnis. Numerische Klassifikation, vol. III, pp. 137–163. INDEKS, Frankfurt (1979)

[5] Borgelt, C.: Resampling for fuzzy clustering. Int. Journal of Uncertainty, Fuzziness and Knowledge-based Systems 15(5), 595–614 (2007)

[6] Chiang, M.M.-T., Mirkin, B.: Experiments for the Number of Clusters in K-Means. In: Neves, J., Santos, M.F., Machado, J.M. (eds.) EPIA 2007. LNCS (LNAI), vol. 4874, pp. 395–405. Springer, Heidelberg (2007)

[7] Davé, R.N.: Characterization and detection of noise in clustering. Pattern Recognition Letters 12, 406–414 (1991)

[8] Duda, R., Hart, P.: Pattern Classification and Scene Analysis. Wiley, New York (1973)

[9] Dunn, J.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Cybernetics and Systems 3(3), 32–57 (1973)

[10] Gustafson, E.E., Kessel, W.C.: Fuzzy clustering with a fuzzy covariance matrix. In: Proc. 18th IEEE Conference on Decision and Control IEEE CDC, San Diego, pp. 761–766. IEEE Press, Turku (1979)

[11] Höppner, F., Klawonn, F.: A contribution to convergence theory of fuzzy c-means and its derivatives. IEEE Transactions on Fuzzy Systems 11, 682–694 (2003)

[12] Höppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy Cluster Analysis. Wiley, Chichester (1999)

[13] Keller, A., Klawonn, F.: Adaptation of cluster sizes in objective function based fuzzy clustering. In: Leondes, C. (ed.) Intelligent Systems: Technology and Applications. Database and Learning Systems, vol. IV, pp. 181–199. CRC Press, Boca Raton (2003)

[14] Klawonn, F.: Fuzzy clustering: Insights and a new approach. Mathware and Soft Computing 11, 125–142 (2004)

[15] Klawonn, F., Höppner, F.: An alternative approach to the fuzzifier in fuzzy clustering to obtain better clustering results. In: Proc. 3rd Eusflat Conference, pp. 730–734 (2003)

[16] Klawonn, F., Höppner, F.: What Is Fuzzy about Fuzzy Clustering? Understanding and Improving the Concept of the Fuzzifier. In: Berthold, M., Lenz, H.-J., Bradley, E., Kruse, R., Borgelt, C. (eds.) IDA 2003. LNCS, vol. 2810, pp. 254–264. Springer, Heidelberg (2003)

[17] Krishnapuram, R., Keller, J.: A possibilistic approach to clustering. IEEE Transactions on Fuzzy Systems 1, 98–110 (1993)

[18] Krishnapuram, R., Frigui, H., Nasraoui, O.: Possibilistic shell clustering algorithms and their application to boundary detection and surface approximation – part 1 & 2. IEEE Transactions on Fuzzy Systems 1, 29–60 (1995)

[19] R Development Core Team (2009) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2009), http://www.R-project.org

[20] Timm, H., Kruse, R.: A modification to improve possibilistic fuzzy cluster analysis. In: Proc. 2002 IEEE Intern. Conf. on Fuzzy Systems (FUZZ-IEEE 2002), pp. 1460–1465. IEEE, Honolulu (2002)

[21] Winkler, R., Klawonn, F., Kruse, R.: Fuzzy C-Means in high dimensional spaces. Fuzzy System Applications 1, 1–17 (2011)

Optimal Control Based on Fuzzy Logic

Kai Michels

Abstract. This paper introduces an algorithm for optimal control, whose first idea was developed in a PhD-thesis under supervision of Rudolf Kruse in the mid of the nineties. In that project, the algorithm was developed theoretically and tested in simulation, while in 2011 a new project was started, where this algorithm shall be applied to a given real-world problem with all the restrictions and additional detail problems that arise in real-world applications. The basic idea of this algorithm is to discretize and bound the state space and to find optimal trajectories from any point in this finite state space to a predefined set point. First, the connection weights between each two points of the discretized state space are estimated, which is based on fuzzy logic. Then, the optimum trajectories are calculated with the help of Dijkstra's algorithm.

1 Dynamical Models as Basis for Controller Design

The first and most important step of controller design is to define, which information is available for design and operation of the controller. If a precise linear model (a set of linear differential equations describing the dynamical behavior of the system) is available and the system is controllable (there are enough actuators to drive the system to any given point in state space), one can access to the entire linear control theory for controller design and operation. Depending on the design algorithm, it can be guaranteed, that the closed-loop system with its transfer behavior from the reference input to the output value is optimal with respect to any given definition of *optimality*.

If the linear model is not precise, which means, that there is a certain range of uncertainty about the parameters of the differential equations, norm-optimal controller design methods can be applied, that guarantee not only stability of the closed-loop

Kai Michels
University of Bremen, Institute for Automation Technology, 28359 Bremen, Germany
e-mail: michels@iat.uni-bremen.de

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 45–59.
springerlink.com

system and sufficiently fast transfer behavior, but also robustness, which means, that even if the real system differs from the model in a given range, stability can be guaranteed.

Also for nonlinear plants (described by nonlinear differential equations) classical control theory offers a great variety of algorithms, that also guarantee stability, optimal transfer behavior and robustness. But due to the large possible structural differences of nonlinear differential equations, in the field of nonlinear plants certain controller design algorithms can be applied only for a certain class or type of nonlinearities. A general controller design method for all types of nonlinear plants would be either too unspecific to apply to a given real plant, or the performance of the resulting controller would be too poor, i.e. the response of the output on a change of the reference input value would be too slow. But to sum up, for special classes of nonlinear plants classical control theory offers excellent solutions like for linear systems.

All of the above mentioned classical controller design approaches have in common, that a more or less precise dynamical model in the form of differential or difference equations must be available. Therefore, they could be called *model-oriented*. But on the other hand, there still exists a numerous number of plants, of which no such model exists.

This is the application field of fuzzy control. Here, only a certain idea about the dynamical behavior of the plant must exist, before one can try to develop a fuzzy controller by definition of fuzzy rules and sets. Normally, some fuzzy rules and sets are used to design a first version of the fuzzy controller. Then, in the closed-loop system, the behavior is tested and improved by adding or changing fuzzy rules and sets. This method could be called *controller-oriented*. As an advantage, this method doesn't require detailed knowledge about control theory. But on the other hand, for non-trivial plants the design is very time-consuming and may get even impossible due to increasing complexity. While one fuzzy rule is clear and easy to understand, a large number of fuzzy rules cannot be surveyed any more. Besides that, there exists no proof of stability for a closed-loop system with such a fuzzy controller, because a proof always requires an analytical model of the plant.

Self-tuning fuzzy controllers have been discussed to solve this problem, but at the end, a self-tuning or adaptation strategy for a non-trivial plant, that guarantees stability, must fit exactly to the plant, which means, it must be based on a rather detailed model of the plant. But if such a model is available, classical controller design approaches lead to much better results.

The same argument holds for using genetic algorithms to find the optimum or at least a good fuzzy controller. Here, each fuzzy controller is seen as an individual of a population, whose quality (fitness) can be estimated in different simulation runs. But to perform a simulation, a precise model of the plant must be available, which leads to the same argument as before.

Neural networks are often seen as solution of this problem, but they just shift the problem into a different area, like classical adaptation algorithms. It is true, that neural networks don't need a plant model, but they need training data, and this

training data must cover the entire range of operation. In practical application on higher-order plants, it is nearly impossible to create such a training data set.

The approach introduced in this paper is a model-oriented approach, where the model is based on fuzzy logic. Due to this fact, the following controller design cannot be analytical, it is based on graph theory. The model information has to be taken from an analytical model, so one might ask, what is the use of this method, as for an existing analytical model classical controller design is normally the better way. But compared to classical controller design methods, the method of this paper doesn't have restrictions regarding the dynamical structure of the plant, it works in principle for any type of differential equations, linear or nonlinear.

In the first step, the state space has to be discretized and bounded, which is normally no problem, as the range of operation is known. Then, for each point in the discretized state space, using the analytical model of the plant it is calculated for different values of the actuating variables, how the neighbouring discrete states can be approximated in one time-step. The distance between the following state and the regarding neighbouring state defines the weight of the graph from the initial state to the neighbouring state. The larger the distance, the smaller the weight.

In this way, for each point in discrete state space it is investigated, how good (close) the system can be brought from this point to its neighbouring points. After this, it is easy to find an optimum trajectory from any point to a given set point in state space using Dijkstra's algorithm.

As the idea of the algorithm is based on fuzzy models and fuzzy logic, we will describe these fuzzy features first.

2 Fuzzy Model

As the entire algorithm can only be realized on a computer, that always needs a certain time to sample the measured values and to estimate a new controller output, the dynamical behavior of the controller as well as of the plant have to be discussed in form of difference equations and not in form of differential equations.

From the measurements known at a certain time $t = kT$ (including past values), a dynamical model of a plant should be able to predict the vector of output variables $\mathbf{y}(k+1)$ or the state vector $\mathbf{x}(k+1)$ at time $t = (k+1)T$. A state space model, for example, describes the relation between the current state vector $\mathbf{x}(k)$, the current vector of actuating variables $\mathbf{u}(k)$ and the change of the state vector in the next time step $\Delta\mathbf{x}(k+1)$ resulting from them. The difference equation for such a model is

$$\Delta\mathbf{x}(k+1) = \mathbf{x}(k+1) - \mathbf{x}(k) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) \tag{1}$$

where $\mathbf{f}$ can be any nonlinear vector function.

An alternative to the state space model is a model, which describes the relationship between actual input vector and former input and output vector on the one hand and the actual output vector on the other hand:

$$\mathbf{y}(k) = \mathbf{f}[\mathbf{u}(k-n), ..., \mathbf{u}(k), \mathbf{y}(k-n), ..., \mathbf{y}(k-1)] \tag{2}$$

In the linear case, such a difference equation corresponds to the well-known transfer function resp. transfer matrix of a system.

In the following, we will use the state space model as given in Eq. (1). Such a model shall now be given in form of a fuzzy model [12, 13]. For reasons of simplicity, we explain a fuzzy model here only for a zero-order SISO (single input - single output) plant with no internal dynamics. The dynamical behavior of such a type of plant can be described by pairs of input and output values $(u(k), y(k))$ at one sample time $t = kT$, where both values are of the set X, which can be seen as the set of real numbers under the restriction of discretization. We therefore have a relation between each value of the actuating variable $u(k)$ and the corresponding one of the output variable $y(k)$, which we can represent by a—still "crisp"—relation $R_k = \{(u(k), x(k))\}$, that means, we just have to store the pair of values $(u(k), y(k))$.

Now we can expect, that, if the pair $(u(k), y(k))$ can appear, similar pairs of measured values can appear as well Using the similarity relation [5, 8, 10]

$$\begin{aligned} E : (X \times X)^2 &\to [0,1], \\ ((u_1, y_1), (u_2, y_2)) &\to \\ \min\{1 - \min(|u_1 - u_2|, 1), 1 - \min(|y_1 - y_2|, 1)\} & \end{aligned} \tag{3}$$

we can replace R_k by a fuzzy relation, the extensional hull of R_k:

$$\begin{aligned} \mu_{R_k} : X \times X &\to [0,1], \\ (u, y) &\to \\ \min\{1 - \min(|u(k) - u|, 1), 1 - \min(|y(k) - y|, 1)\} & \end{aligned} \tag{4}$$

The point $(u(k), y(k))$ in the $u - y-$ plane becomes the fuzzy set μ_{R_k} (cf. Fig. 1). From a set-theoretical point of view, μ_{R_k} is the set of all points which are similar to $(u(k), y(k))$, where "similarity" is defined by Eq. (4).

We can also think of this fuzzy relation as the set of all pairs of values (u, y) which are possible for this plant. The pair $(u(k), y(k))$, as a pair of values which were given in the beginning, maybe by measurement, is certainly possible, and its degree of membership to this set is therefore 1. For other pairs of values, the degree of membership to this set decreases with an increasing distance from $(u(k), y(k))$. The assumption is, that pairs, which are close to the measured pair of values, are also possible; in fact, they are more possible, the smaller the distance to this pair is. It should be mentioned that we would obtain a different fuzzy relation μ_{R_k}, if we chose another similarity relation. With regard to the computational effort however, it seems reasonable to choose a relation which is as simple as possible for the construction of the model.

The disjunctive combination of a certain number of fuzzy relations μ_{R_k} delivers the fuzzy model of the plant:

$$\mu_R = \bigcup_k \mu_{R_k} \tag{5}$$

Figure 1 shows such a model as a result of two pairs of values $(u(1), y(1))$ and $(u(2), y(2))$. The more pairs of values form the model, the more it will cover the field of operation.

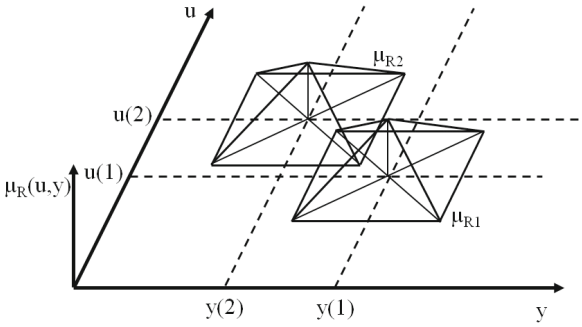


Fig. 1 Fuzzy model of a plant derived from two pairs of values

As a firm footing for our further explanations, let us now describe how to compute the expected output value $y_m(k)$ for a given input value $u_m(k)$ using this model. First, we have to define a singleton fuzzy set representing the input:

$$\mu_u : X \to [0,1], u \to \begin{cases} 1 & \text{if } u = u_m(k) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Then, using this singleton and the given relation describing the model μ_R, we compute the relational equation $\mu_y = \mu_u \circ \mu_R$. We obtain a fuzzy set as the output quantity:

$$\begin{aligned} &\mu_y : X \to [0,1], \\ &\quad y \to \sup\{\min[\mu_u(u), \mu_R(u,y)] \mid u \in U\} = \mu_R(u_m(k), y) = \mu_y(y) \end{aligned} \tag{7}$$

This procedure corresponds to making a cut parallel to the y-axis through the relation μ_R at $u = u_m(k)$, and projecting it onto the output variable y (see Fig. 2). A defuzzification of this fuzzy set yields the value which should be expected as the output value $y_m(k)$. These are obviously the same steps which have to be carried out to calculate the output value of a conventional fuzzy controller, if the rules are stored in form of a fuzzy relation.

The entire method can be adjusted to MIMO (multi input - multi output) plants of higher order without any problems. A first order plant, for example, can be characterized by a triple of measured values $(u(k), x(k), \Delta x(k+1))$ (actuating variable, state, resulting change of state), where the actuating variable and the state form the input quantities of the model, and the resulting change of state is the output

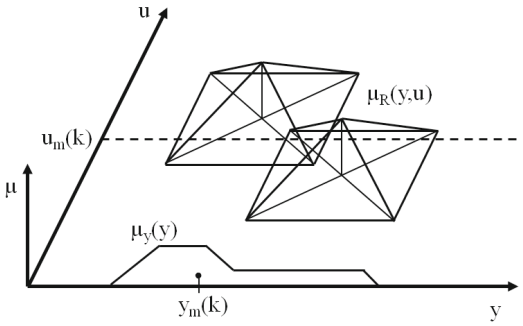


Fig. 2 Computing the output value of a fuzzy model

quantity. Therefore, the fuzzy model requires an additional dimension. Normally we may obtain multi-dimensional fuzzy models. But as any of the equations involved can easily be extended to any dimension, the method itself is not affected by adding further dimensions.

Let us discuss this type of modelling from a logical point of view: The more information is added to the model (by adding new pairs of values with their fuzzy relations), the "larger" the fuzzy set gets. If we got the model from pairs of measured values, which might be noisy, we might get several pairs of values, that have slightly different output values for the same input value. This would lead to a quite "uncrisp" relation describing the plant behavior. At the end we would have to face the fact that adding more information leads to a less precise and therefore worse model of the plant.

Using conjunctive connected implications (Goedel, Lukasiewicz) instead of disjunctive connected similarity relations for modelling would cause the opposite effect: For every new pair of measured values the relation R would become smaller and sharper, which means, the model gets more exactness and precision. Indeed this is the usual intention for adding new information to the model. But if the input data for modelling is noisy, each conjunctive connection of two implications resulting from similar but inconsistent pairs of measured values would lead to deletion of each others information, so that at the end the final overall implication could be zero everywhere. Because of that effect, we prefer the disjunctive connection of similarity relations for our work.

It should be mentioned how such a model can be stored. It seems reasonable to discretize the entire space spanned by the involved quantities. At every supporting point which results from this process, the degree of membership valid at that point is recorded (see Fig. 3). The relation describing the model is then defined in terms of an interpolation between these recorded degrees. Of course, this causes differences between the stored relation and the original one, but we can adjust this difference according to our needs by choosing a grid of sufficiently high resolution.

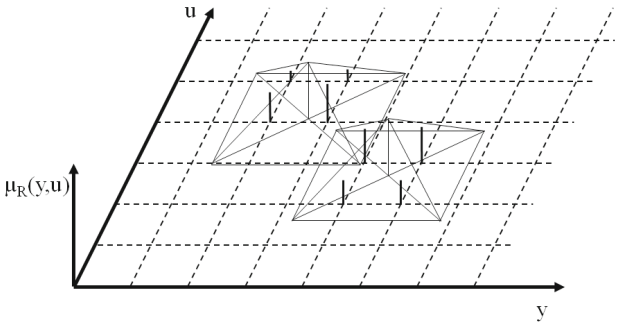


Fig. 3 Illustration of how to store a fuzzy model

3 Model-Based Control

Obviously, interchanging input and output while using a given fuzzy model (see Fig. 2) would cause no problem. Therefore, we can use the model to calculate the corresponding input value for a given output value, i.e. to answer the question, which actuating value (or vector) is necessary to drive the system to a certain state [6, 7]. From a theoretical point of view, we use the inverted model, which is well known as compensating control in classical control theory. But while inversion of an analytical model causes problems in most practical applications, the inversion of a fuzzy model is just interchanging input and output of the model. We only have to define the plant output we want, get into the model and calculate the corresponding input of the plant.

As it is easy to construct fuzzy relations also for multi-dimensional systems without any difficulties, this type of model-inversion should cause no problems for higher order plants or MIMO systems. The models are constructed in the same way as usual, and then just used invertedly.

If we have got a state-space fuzzy model, i.e. Eq. (1), we have to define the desired change of the state vector $\Delta\mathbf{x}(k+1)$ at the next time step as input for our model to get the necessary actuating (or input) vector $\mathbf{u}(k)$ at the current time step. But here, we face a problem. If the change of state we want is too large, there may exist no input vector that can cause this change. We therefore have to split the wanted change of state into several smaller changes and perform the entire procedure for several times. To do so, we need a block in front of the inverted model to calculate these intermediate values (see Fig. 4).

But for calculation of intermediate values, we face another problem, that can easily be explained in form of an example. Let us assume we are given a mass m to move from one point to another in a plane. The differential equations are

$$\begin{aligned} F &= ma \\ a &= \dot{v} \\ v &= \dot{l} \end{aligned} \tag{8}$$

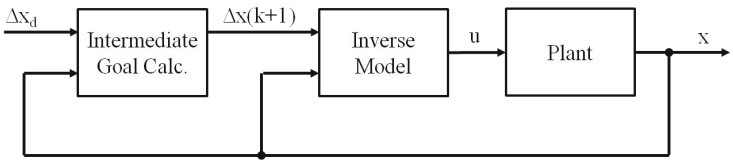


Fig. 4 Control based on an inverse model

where F is the driving force, a is acceleration, v is velocity and l is position. The state variables are l and v. We would like to achieve a transformation of the system from the initial state $(l_1, v_1) = (0,0)$ into the final state $(l_3, v_3) = (w,0)$ with $w > 0$. Figure 5 shows the required trajectory in state space. To get from state 1 to state 3, we obviously need an intermediate velocity $v > 0$ in order to bring the mass to its final state, i.e. we have to go via state 2.

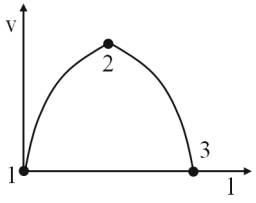


Fig. 5 Trajectory of moving a mass

A simple calculation of intermediate goals would just have estimated the direct connection between initial and final state and splitted this connection into several smaller steps, so that the system could follow each step within one time-step. All intermediate reference values would have been on the l-axis, which means, v would always have been zero. This would be equal to the demand, that the body moves from position $l = 0$ to $l = w$ with a velocity of zero, which is obviously impossible.

The example shows clearly, that the calculation of intermediate values requires knowledge about the dynamical behavior of the plant. This leads to the idea of optimal control based on fuzzy logic.

4 Optimal Control

This method calculates the optimal trajectory from the initial to the final state first. Following this trajectory the intermediate reference values (as part of this trajectory) are then calculated for each time-step and given as input to the inverse model of the plant. As the calculation of the optimal trajectories requires a lot of computation time, this method is suitable especially for systems with a constant reference value. For a continuously varying reference value, however, the computational effort might exceed reasonable limits.

Let us explain the calculation of optimal trajectories for a second order system with one actuating variable and a fixed reference value or final state. The model-relation μ_R has got the two states $x_1(k)$ and $x_2(k)$ and the actuating variable $u(k)$ as input quantities. The output quantities are given by the changes $\Delta x_1(k+1)$ and $\Delta x_2(k+1)$ of the states during the following time step, resulting from the input quantities. μ_R is therefore a five-dimensional relation.

First, we have to choose an operating region around the final state, where we are sure of that the system will never leave it. We discretize this limited state space, which leaves us with a finite number of discrete states inside this region. Figure 6 shows such a discretization for a second order system. The origin of the coordinate system is assumed to be the final state.

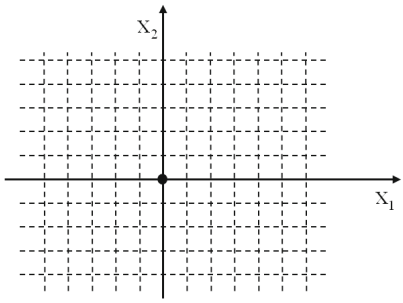


Fig. 6 Discretization of a two-dimensional state space

As the second step, for every single state we have to detect the possibility with which it can be transferred into one of its adjacent states by means of a suitable actuating variable, within one time step. Figure 7 gives an illustration of this for one state of our example. The central point is the state under examination. It is surrounded by eight adjacent states. We are now looking, for example, for the possibility with which this state may be transferred into the state on the upper right. We know the coordinates (x_{1m}, x_{2m}) of the central state, and those of the state on the upper right, (x_{1r}, x_{2r}). We then define the state in the center to be the current state $(x_1(k), x_2(k)) = (x_{1m}, x_{2m})$, and the difference between this state and the one on the upper right $(\Delta x_1(k+1), \Delta x_2(k+1)) = (x_{1r} - x_{1m}, x_{2r} - x_{2m})$ to be the desired state difference for the next sample.

We then have to define membership functions for $x_1(k), x_2(k), \Delta x_1(k+1)$ and $\Delta x_2(k+1)$ as inputs for our fuzzy model to calculate the corresponding actuating value $u(k)$. As $x_1(k), x_2(k)$ represent the well known and precisely defined current state of the system, we use singletons for these variables. We could also use singletons for $\Delta x_1(k+1), \Delta x_2(k+1)$, but here it makes more sense to define the membership functions by [4]

$$\mu_{\Delta x_i} : X \to [0,1], \Delta x_i \to \begin{cases} \frac{\Delta x_i}{\Delta x_i(k+1)} & \text{if } 0 < \Delta x_i < \Delta x_i(k+1) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

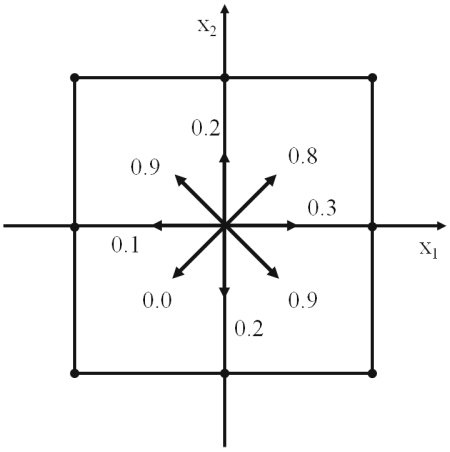


Fig. 7 Possibilities for a transition from one state to one of its adjacent states

This membership function is shown in Fig. 8. This way to define the membership function makes sense because of the following reason: It might happen, that no actuating value exists to drive the system within one time-step from the central state to the upper right one. In that case, using singleton membership functions for $\Delta x_1(k+1), \Delta x_2(k+1)$ would lead to no result from the fuzzy model. This case is prevented by using membership functions as shown here. Even if no actuating value can drive the system into the upper right state within a single time-step, we will at least find an actuating value that drives the system as close as possible to this state.

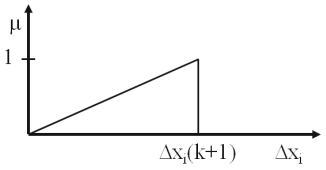


Fig. 8 Membership function for $\Delta x_i(k+1)$ as fuzzy model input

Using the fuzzy sets defined above and the model-relation μ_R, we can then estimate the output of the relational equation

$$\mu_u = \mu_{x_1} \circ \mu_{x_2} \circ \mu_{\Delta x_1} \circ \mu_{\Delta x_2} \circ \mu_R \tag{10}$$

which leaves us with a fuzzy set μ_u for the actuating variable. This fuzzy set contains all actuating values that can drive the system from the center state closer to the upper right state.

A defuzzification is not necessary, since we are only interested in a measure of the possibility of the transition from one state to another one—and not in the

actuating variable which would be required to obtain this transition. This measure is the highest degree of membership

$$P = \sup_{u} \{\mu_u(u)\} \tag{11}$$

which occurs within the fuzzy set μ_u (0.8 in our example). It would make little sense to use a measure like for example $\int \mu_u(u)du$, since we are not interested in the power of the set, but only in whether there actually exists an actuating variable which can be used to enforce the transition from the center state to the upper right one.

As the result of this step, we now know the possibilities for all the transitions from any state of the limited, discretized state space to any of its adjacent states that can occur within one time step. We can think of this as having two directed lines connecting any two adjacent states, which are labeled with a measure describing the possibility of the corresponding state transition. In Fig. 7 only the lines beginning at the center state are sketched.

Now, we would like to find one trajectory from every point of the discretized state space into the final state, passing through several other discrete states if necessary. On the one hand, we would like this trajectory to be as short as possible. On the other hand, it should contain only those transitions which have a high value of possibility, since this increases the possibility that the real system can actually follow this trajectory. In order to be able to present this task as an optimization problem of a closed form, we transform the possibility values of all the transitions according to

$$P' = 1.0 - P^a \qquad \text{where } a > 0 \tag{12}$$

The value for P' will be the smaller, the higher the possibility for the corresponding transition is. a can be used to manipulate the relation of the P'-values of large to small possibility values. This again affects the probability of having transitions with small or large possibility values occur within the computed trajectories.

If all the possibility values are transformed, we can define the weight of a trajectory to be the sum of all P'-values of the transitions involved. This weight will be the smaller, the fewer transitions the trajectory contains and the smaller the values P' of the corresponding transitions are. Now this just means that the trajectory will be short and contain transitions with high possibility values. Our task is therefore to find a trajectory from every point of the state space to the final state with a weight as small as possible.

This is a task for *Dijkstra's Algorithm* [2] which is well known in graph theory and we are now going to present shortly. It is defined recursively, which is why we have to presuppose a connected domain of the discretized state space, where all the optimal trajectories from any state to the final state, which lie completely inside this region, are already known. We now want to add another (adjacent) state to this region, i.e. the optimal trajectory inside this region has to be computed for the new state. As the state is adjacent to the region, some transitions between this state and states inside the region will exist, with corresponding weights P'.

Since we already know the optimal trajectory of each of these adjacent old states, all we have to do is find the state whose trajectory is also optimal from the new state via this state to the final state. And this merely requires adding the weight of the transition from the new to the old state to the weight of the optimal trajectory from the old state to the final state. The optimal trajectory for the new state will be the one passing through that old state, for which the addition produced the smallest value. We then have to store that old state as successor of the new state, and also store the weight of the new state's optimal trajectory.

Having done that, we also have to check if adding this new state changes the optimal trajectory of an old state, i.e. if it is "cheaper" for any of the old states to reach the final state via this new state. For all of the old states, for which this holds, we have to store the new state as successor and change the weight of their optimal trajectory. Then, we have finished one recursion, add the new state to the (extended) region, and repeat the entire algorithm for the next new state all over again.

Let us illustrate this algorithm with the simple example given in Fig. 9. The final state is S. The region for which we already know all the optimal trajectories consists of A, B and S. We would like to add N as the new state. We know the weights for the transitions between N and its neighbors A and B in both directions. The weight for the trajectory $N-A-S$ is $0.3+0.4=0.7$, the one for $N-B-S$ is $0.1+0.9=1.0$. The optimal trajectory from N to S therefore runs via A. We see that the weight for the trajectory $B-N-A-S$ $(=0.8)$ is less than the one of the originally optimal trajectory from B to S $(=0.9)$. We therefore define a new optimal trajectory for B, too, which now runs from B to S via N and A. At the end of this recursion step, we define A as the successor of N, and N as the successor of B.

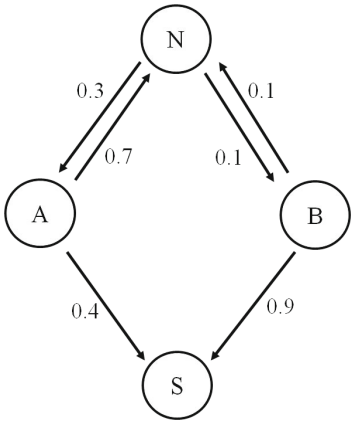


Fig. 9 An illustration to Dijkstra's algorithm

With Dijkstra's algorithm, we can now estimate the optimal trajectories from all the discrete states of the limited state space to the final state, starting from this final state (see Fig. 10). With the help of these trajectories, we can always find a suitable intermediate reference value for the control.

The algorithm for any time step is the following: We have to measure the current state of the plant, and determine the discrete state which is closest to it. The successor state of this state with respect to the computed trajectory is then used as the intermediate reference value for the inverse model. From the inverse model we get the necessary actuating variable to drive the system to the intermediate reference value within the next time step at least approximately. The state reached at the end of this time step is the starting point for the algorithm at the next time step.

The question arises, how far the only approximative reaching of the intermediate reference value within one time step affects the entire control algorithm, as this will be the normal case. Through accumulation of these errors, it is even possible that the system continues to move away from the computed trajectory, and that it finally reaches a state which is closer to another computed trajectory. However, this does not effect the overall result, as all that happens is that the system then reaches the final state via this other trajectory.

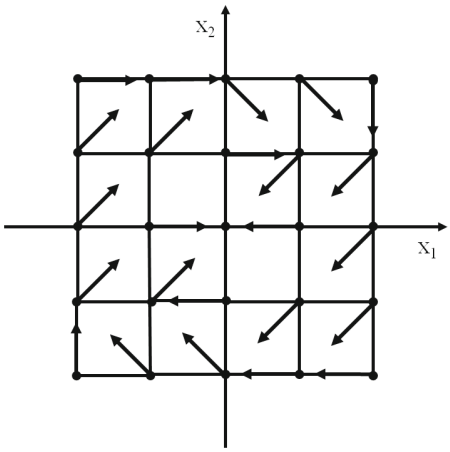


Fig. 10 Trajectories in a discretized, limited state space

5 Practical Remarks

In principle, the fuzzy model of the plant can be generated from measured values. Any tuple of measurements, e.g.

$$(u(k), x_1(k), x_2(k), \Delta x_1(k+1), \Delta x_2(k+1)) \tag{13}$$

for a second-order system as described in section 4, forms a partial fuzzy relation μ_{R_k}, that has to be connected disjunctively to the other partial fuzzy relations resulting from other measurement tupels, so that at the end the resulting overall fuzzy relation represents the plant's behavior in the entire range of operation.

But even for just a second-order plant this is nearly impossible, because it would require, that during an identification phase the plant is driven through the entire state plane, and furthermore, that for all states all different actuating (input) values have been applied.

Therefore it makes more sense to derive the fuzzy model from a given analytical model of the plant. We can use this model to calculate for any state and any input (actuating) vector the resulting output, which leads to one measurement tuple, that forms a partial fuzzy relation to be connected to the overall fuzzy relation.

But doing this, the question arises why we should transfer an analytical model into a fuzzy model instead of using the analytical model directly for trajectory generation. Indeed, this is done in our new research project. For each discrete state we calculate the resulting following state for different values of the actuating vector. So we check, how close we can get to one of the neighbouring states. The smallest distance to the neighbouring state leads to the required measure of possibility to reach this neighbouring state within one time-step.

A second problem arises from the huge amount of storage needed for the described algorithm. We have to discretize the state space with such small intervals, that we can drive the system from one discrete state into one of its adjacent states within one time-step. This may lead to more than one hundred discretization intervals for each state variable, so that even for just the control of an inverted pendulum, which is a fourth order system, we would get more than $100^4 = 100,000,000$ points in the discrete state-space.

To reduce this huge amount of storage needed, we have to choose a lower discretization, which means, less discrete points and larger distances between them. But this causes the problem, that no actuating vector can drive the system from one discrete state close to one of its adjacent states. The solution is, that we have to simulate the possible system's behavior under different actuating vectors not only for one single time-step, but for several time-steps, to check how close we can get to the neighbouring states of one given state. The idea of simulating the plant's behavior for several time-steps is well known in classical control theory and referred to as "predictive control", but there, the simulation is used for the entire time interval needed to reach the final state, not only to get from one discrete state to one of its adjacent states. Related approaches based on fuzzy systems can also be found in [1, 3, 9] and [11].

One can see, that these solutions to meet practical requirements forced us to leave the pure fuzzy approach. Instead of fuzzy sets and fuzzy models, we just handle characteristic fields. And due to the increasing discretization distances, the solution can only be suboptimal, but not optimal any more. But the original fuzzy method, to check the possibility to get from one discrete state to another, and to calculate an optimal trajectory based on these possibilities, is still the basis of this method.

Therefore, our algorithm might be an example for a successful combination of classical control methods and fuzzy logic. This should be a good result, if we remember the "war" between fuzzy researchers and classical control engineers in the nineties of the last century.

Acknowledgements. The author would like to thank Prof. Rudolf Kruse for co-supervision of his PhD thesis, which was the basis of the above described project, for supporting the author's academic career in many ways, and finally, for many good times on different conferences all over the world.

References

[1] Babuška, R., Verbruggen, H.B.: Fuzzy modeling and model-based control for nonlinear systems. In: Jamshidi, M., Titli, A., Zadeh, L., Boverie, S. (eds.) Applications of Fuzzy Logic: Towards High Machine Intelligence Quotient Systems, pp. 49–74. Prentice-Hall, Inc., Upper Saddle River (1997)
[2] Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. MIT Press and McGraw-Hill, Cambridge (1990)
[3] Filev, D., Angelov, P.: Fuzzy optimal control. Fuzzy Sets Syst. 47(2), 151–156 (1992)
[4] Gorzalczany, M.B.: Interval-valued fuzzy controller based on verbal model of object. Fuzzy Sets Syst. 28(1), 45–53 (1988)
[5] Kruse, R., Gebhardt, J., Klawonn, F.: Foundations of Fuzzy Systems, 1st edn. John Wiley & Sons, Ltd., Chichester (1994)
[6] Sm, L., Sh, H.: A method of generating control rule model and its application. Fuzzy Sets Syst. 52(1), 33–37 (1992)
[7] Michels, K.: A model-based fuzzy controller. Fuzzy Sets Syst. 85(2), 223–232 (1997)
[8] Michels, K., Klawonn, F., Kruse, R., Nürnberger, A.: Fuzzy Control: Fundamentals, Stability and Design of Fuzzy Controllers. STUDFUZZ, vol. 200. Springer, Heidelberg (2006)
[9] Valente de Oliveira, J., Lemos, J.M.: Long-range predictive adaptive fuzzy relational control. Fuzzy Sets Syst. 70(2-3), 337–357 (1995)
[10] Pedrycz, W.: Fuzzy Control and Fuzzy Systems. Control theory and applications studies series, vol. 3. Research Studies Press, Tauntun (1989)
[11] Sastry, V.N., Tiwari, R.N., Sastri, K.S.: Dynamic programming approach to multiple objective control problem having deterministic or fuzzy goals. Fuzzy Sets Syst. 57(2), 195–202 (1993)
[12] Tong, R.M.: Synthesis of fuzzy models for industrial processes: some recent results. Int. J. General Syst. 4(3), 143–162 (1978)
[13] Xu, C., Lu, Y.: Fuzzy model identification and Self-Learning for dynamic systems. IEEE Trans. Syst. Man Cybern. 17(4), 683–689 (1987)

Kernel Based Defuzzification

Thomas A. Runkler

Abstract. Defuzzification converts a fuzzy set to a crisp value or set. Standard defuzzification methods are the center of gravity (COG) and the mean of maxima (MOM). A popular parametric defuzzification method is based on the basic defuzzification distribution (BADD). COG and MOM are special cases of BADD for unit and infinite exponents, respectively. Kernelization is a popular approach to improve data processing methods by implicit transformation to higher dimensions. This article introduces kernelized defuzzification. We present a kernelized version of COG and illustrate it for polynomial kernels (pkCOG) and Gaussian kernels (GkCOG). We show that pkCOG is equivalent to BADD. Experiments with various representative synthetic examples show that GkCOG is superior to pkCOG/BADD in terms of smoothness.

1 Introduction

Fuzzy systems perform computation and reasoning based on fuzzy sets [46]. Inputs for fuzzy systems are mostly crisp, but also some applications with fuzzy inputs have been reported [24]. Fuzzy computation and reasoning mostly produces fuzzy results, but in many applications crisp results are required, so the fuzzy results have to be converted into crisp results. The conversion of fuzzy sets into crisp sets or values is called *defuzzification*.

Defuzzification is a very active research field since the early 1990s. Standard defuzzification methods are the *centroid* or *center of gravity (COG)* method that computes the centroid (first order moment) of the area under the membership function, and *maxima* methods that yield one of the values that has maximum membership, for example the *mean of maxima (MOM)*. Many variations of these standard

Thomas A. Runkler
Siemens AG, Corporate Technology, 81739 Munich, Germany
e-mail: thomas.runkler@siemens.com

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 61–72.
springerlink.com

defuzzification methods have been proposed. The *centroid of largest area (COLA)* considers only the largest sub-area under the membership function [26, 44]. The *center of area (COA)* or *median* computes the median instead of the first order moment. Since the determination of the first order moment is computationally intensive and infeasible for applications with severe real-time constraints, various efficient algorithms for the computation of COG [2, 16, 25, 43] and also implementations of COG devices in dedicated hardware modules [1, 19, 20, 39] have been proposed. The *decreased effort centroid defuzzification algorithm (DECADE)* [32] is an efficient algorithm that approximates the COG. Instead of the *mean* of maxima (MOM), also the *first of maxima (FOM)*, *last of maxima (LOM)*, and *center of maxima (COM)* (median) are used. More sophisticated choices of the maxima are presented in [15]. Defuzzification has also been interpreted as a decision process, which yields the *constrained decision defuzzification (CDD)* [34], multi-criteria decision methods [23], and constrained decision methods [21]. Also measurement-theory has been applied to defuzzification [33]. Defuzzification can also be done by modern methods of computational intelligence, such as clustering [7, 40], neural networks [14, 27, 10], or evolutionary algorithms [3, 13]. The previously reported defuzzification methods transform a fuzzy set into only one crisp value, but also *subset defuzzification* methods have been proposed that yield crisp *sets* [35, 41]. Recently, also the defuzzification of type-2 fuzzy sets has been considered in the literature [4, 5, 8, 9, 17, 18, 38, 42]. For a comprehensive overview of defuzzification methods see [28, 29], and for specific issues concerning fuzzy control see [12, 36].

This article does not focus on individual defuzzification methods but on parametric defuzzification that covers standard defuzzification methods as special cases, and whose properties can be determined by setting appropriate parameter values. A widely used class of parametric defuzzification methods is based on the *basic defuzzification distribution (BADD)* [6]. BADD contains COG as a special case but can also approximate MOM. A linear approximation of BADD with lower computational effort is the *semi linear defuzzification (SLIDE)* [45], and *extended center of area (XCOA)* is a BADD variant that uses the median instead of the center of gravity [31].

Based on Mercer's theorem from 1909 [22], the so-called *kernel trick* has recently become popular in many fields of data processing such as *support vector machines* for classification [37] or *kernelized clustering* [11, 30]. The idea of kernelization is to transform the data to a very high-dimensional (possibly infinite-dimensional) space where they have more desirable properties so that problems can be solved more appropriately. For example, nonlinear class borders can be approximated by linear class borders in the high-dimensional space, and the effect of outliers is reduced in the high-dimensional space which enables more robust clustering algorithms. In this article we propose a kernelized version of COG defuzzification that contains COG and BADD as special cases. Experiments with representative synthetic examples show that the proposed kernelized clustering is superior to COG and BADD.

This article is structured as follows: In section 2 we briefly review the standard defuzzification methods such as COG. In section 3 we present BADD as a popular example for parametric defuzzifation. In section 4 we introduce the kernelized COG defuzzification. In section 5 we illustrate kernelized clustering in experiments with representative examples and compare kernelized COG with regular (unkernelized) COG and BADD. In section 6 we summarize our conclusions.

2 Standard Defuzzification Methods

We denote a fuzzy set over a universe $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p$, $n, p \in \mathbb{N}$, as a set $U = \{u_1, \ldots, u_n\} \subset [0,1]$, where each u_k quantifies the membership of x_k in the fuzzy set. Fuzzy sets are often defined over one-dimensional universes, $p = 1$, and multi-dimensional fuzzy sets can be constructed from individual one-dimensional fuzzy sets (e.g. using t-norms of the cylindrical extensions). In our experiments we will therefore also consider only one-dimensional examples, but all presented methods are also valid for higher-dimensional fuzzy sets, $p \in \mathbb{N}$. Using these definitions, the *centroid* or *center of gravity (COG)* can be defined as the first moment of the area under the membership function

$$y = \frac{\sum_{k=1}^{n} u_k x_k}{\sum_{k=1}^{n} u_k} \tag{1}$$

The *mean of maxima (MOM)* is defined as

$$y = \frac{\sum_{k \in \operatorname{argmax} U} x_k}{\sum_{k \in \operatorname{argmax} U} 1} \tag{2}$$

where $\operatorname{argmax} U$ is the index set of the maxima in U. Notice that for both COG and MOM we have $y \in \mathbb{R}^p$, but not necessarily $y \in X$.

3 Parametric Defuzzification

As a generalization of COG (1), Filev and Yager introduced the *basic defuzzification distribution (BADD)* [6]. The BADD defuzzification is defined as

$$y = \frac{\sum_{k=1}^{n} u_k^s x_k}{\sum_{k=1}^{n} u_k^s} \tag{3}$$

with the parameter $s > 0$. Obviously, COG is a special case of BADD for $s = 1$. For $s \to \infty$ we obtain

$$\lim_{s\to\infty} \frac{\sum_{k=1}^{n} u_k^s x_k}{\sum_{k=1}^{n} u_k^s} = \lim_{s\to\infty} \frac{(\max U)^k \sum_{k=1}^{n} \left(\frac{u_k}{\max U}\right)^s x_k}{(\max U)^k \sum_{k=1}^{n} \left(\frac{u_k}{\max U}\right)^s} \tag{4}$$

$$= \lim_{s\to\infty} \frac{\sum_{k\in \operatorname{argmax} U} \left(\frac{\max U}{\max U}\right)^s x_k}{\sum_{k\in \operatorname{argmax} U} \left(\frac{\max U}{\max U}\right)^s} = \frac{\sum_{k\in \operatorname{argmax} U} x_k}{\sum_{k\in \operatorname{argmax} U} 1} \tag{5}$$

which is equivalent to MOM (2). So for large values of s, BADD approximates MOM.

4 Kernel Based Defuzzification

Mercer's theorem [22] states that for any data set X and any so-called *kernel function* $\kappa : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ there is a mapping $\varphi : \mathbb{R}^p \to \mathbb{R}^q$, $q \in \mathbb{N}$, so that

$$\kappa(x_j, x_k) = \varphi(x'_j) \cdot \varphi(x'_k)^T \tag{6}$$

This means that a mapping of X to X' can be implicitly done by replacing scalar products in X' by kernel functions in X, without explicitly computing X'. This is called the *kernel trick*. Frequently used kernel functions include

$$\text{linear kernel } \kappa(x_j, x_k) = x_j \cdot x_k^T \tag{7}$$

$$\text{polynomial kernel } \kappa(x_j, x_k) = (x_j \cdot x_k^T)^s, \quad s > 0 \tag{8}$$

$$\text{Gaussian kernel } \kappa(x_j, x_k) = e^{-\frac{\|x_j - x_k\|^2}{s}}, \quad s > 0 \tag{9}$$

Usually, we assume $q \gg p$, i.e. the data are mapped to a higher dimensional space, but since the transformation is not explicitly performed, the value of q can be ignored here.

In this article we do not want to kernelize the data X but we want to kernelize the memberships U in COG. Since COG (1) does not contain any scalar product in U, we can not immediately apply the kernel trick. Instead we define the singleton set

$$\delta^k = \{\delta_1^k, \ldots, \delta_n^k\} \tag{10}$$

with $k \in \{1, \ldots, n\}$, where

$$\delta_j^k = \begin{cases} 1 & \text{if j=k} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

with $j \in \{1,\ldots,n\}$, so we can write

$$u_k = \sum_{j=1}^{n} \delta_j^k \cdot u_j = \delta^k \cdot u^T \tag{12}$$

for δ^k and u in vector form. We can therefore kernelize u_k as

$$u_k = \kappa(\delta^k, u) \tag{13}$$

Inserting the kernelized u_k into COG (11) yields

$$y = \frac{\sum_{k=1}^{n} k(\delta^k, u)x_k}{\sum_{k=1}^{n} k(\delta^k, u)} \tag{14}$$

If we use the polynomial kernel (8), then we obtain

$$y = \frac{\sum_{k=1}^{n} k(\delta^k, u)x_k}{\sum_{k=1}^{n} k(\delta^k, u)} = \frac{\sum_{k=1}^{n} (\delta^k \cdot u^T)^s x_k}{\sum_{k=1}^{n} (\delta^k \cdot u^T)^s} = \frac{\sum_{k=1}^{n} u_k^s x_k}{\sum_{k=1}^{n} u_k^s} \tag{15}$$

which is equal to the BADD defuzzification (3). So, for polynomial kernels, kernelized COG is equal to BADD.

A very frequently used kernel function is the Gaussian kernel (9). If we use the Gaussian kernel (9) in kernelized COG (14), then we obtain what we call *Gaussian kernel COG (GkCOG)*.

$$y = \frac{\sum_{k=1}^{n} x_k e^{-\frac{\|\delta^k - u\|^2}{s}}}{\sum_{k=1}^{n} e^{-\frac{\|\delta^k - u\|^2}{s}}} \tag{16}$$

Notice that the kernelization does not imply any computational benefit but yields some nice functional properties, as we will see in the next section.

5 Experiments

We present four different experiments with synthetic data to examine and compare the behaviors of COG, BADD, and GkCOG defuzzification. For each experiment for both BADD and GkCOG we use the parameters $s = \{0.01, 0.0125, 0.016, 0.02, 0.025, 0.032, 0.04, 0.05, 0.063, 0.08, 0.1, 0.125, 0.16, 0.2, 0.25, 0.32, 0.4, 0.5, 0.63, 0.8, 1, 1.25, 1.6, 2, 2.5, 3.2, 4, 5, 6.3, 8, 10, 12.5, 16, 20, 25, 32, 40, 50, 63, 80, 100\}$ which represent an approximately logarithmic scaling over 4 orders of magnitude. In each experiment we consider the one dimensional data set $X = \{0, 0.1, \ldots, 1\}$.

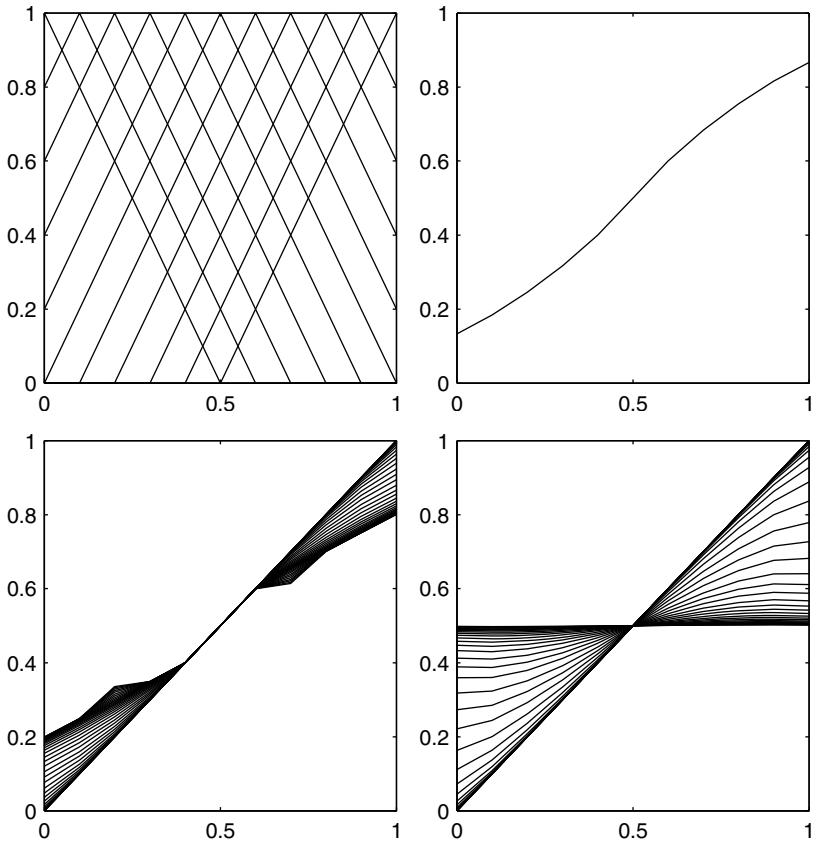


Fig. 1 Experiment 1: truncated triangles with various peak positions, COG (top right), BADD (bottom left), and GkCOG (bottom right)

As pointed out earlier, we restrict to one-dimensional data here because they are most common and because of the easier visualization. However, the presented methods also work for higher-dimensional data.

In our first experiment we consider triangular membership functions with width 1, height 1, and variable peaks at $0, 0.1, \ldots, 1$, which are truncated at $x = 0$ and $x = 1$. Fig. 1 (top left) shows the 11 different membership functions generated this way. For each peak position $0, 0.1, \ldots, 1$ we compute the defuzzified value. Fig. 1 (top right) shows the results for COG (vertical) over the center position (horizontal). For a triangle with peak at 0, COG yields about 0.15, at 0.5 we have 0.5, and at 1 we have about 0.85. Fig. 1 (bottom left) shows the results for BADD. For large values of s, BADD behaves like MOM and follows the peak of the triangle, which yields the main diagonal. For $s = 1$, BADD is equal to COG and yields the same curve as in Fig. 1 (top right). For $1 < s < \infty$, BADD yields curves between MOM and BADD. For $s < 1$ BADD yields some unexpected artefacts (bumps at about 0.2

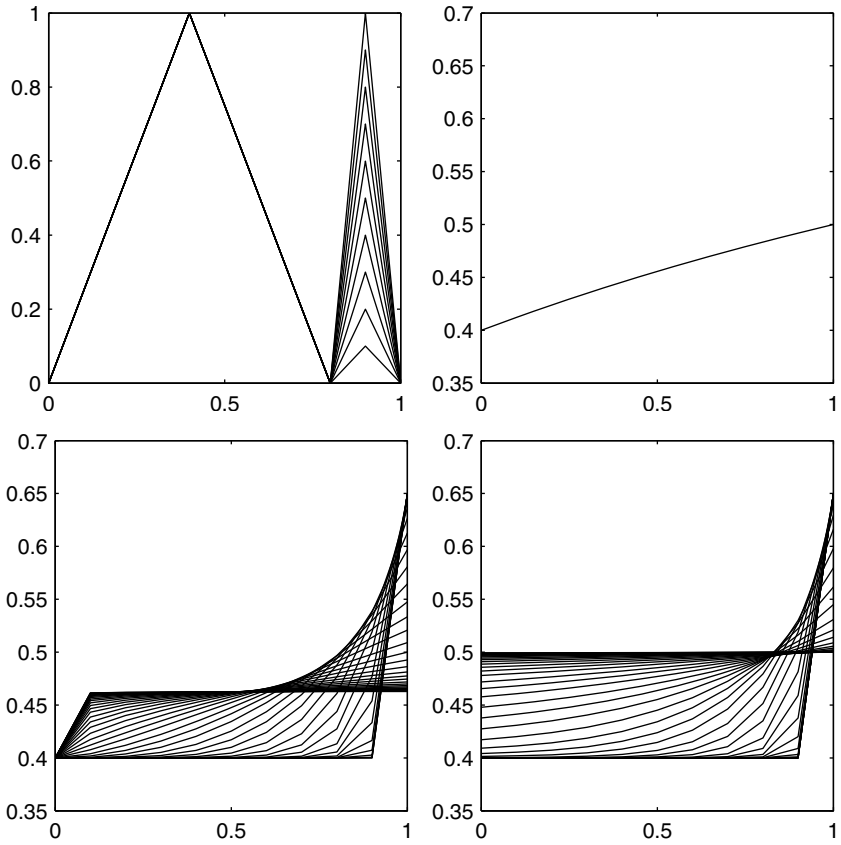


Fig. 2 Experiment 2: wide triangle and narrow triangles with various heights, COG (top right), BADD (bottom left), and GkCOG (bottom right)

and 0.8). For triangle peaks between 0.4 and 0.6, BADD yields the same values for any parameter s which is somewhat counterintuitive. Fig. 1 (bottom right) shows the results for GkCOG. Also here we can see the main diagonal representing MOM as one extreme. However, here the other extreme is a constant defuzzification value of 0.5 representing ignorance of the provided fuzzy information. GkCOG smoothly interpolates between these two extremes. There are no bumps, and the choice of s affects all results except for the symmetric case of a triangle peak at 0.5 which matches intuitive expectation.

In our second experiment we consider one triangle with peak 0.4, width 0.8, height 1, and another narrower triangle with peak 0.9, width 0.2, and variable height $0, 0.1, \ldots, 1$ (Fig. 2, top left). For each height $0, 0.1, \ldots, 1$ we compute the defuzzified value. Fig. 2 (top right) shows the results for COG (vertical) over the height (horizontal). For height 0, COG yields 0.4 (the mean of the wide triangle), for height 1, COG yields 0.5, and approximately linearly interpolates between the two

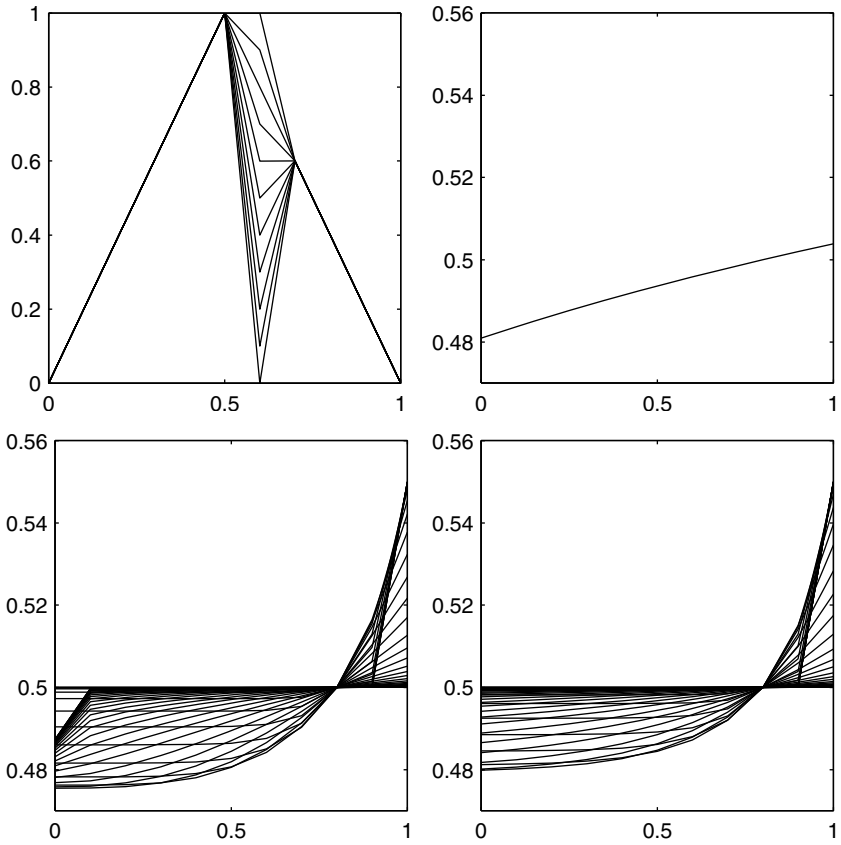


Fig. 3 Experiment 3: triangle with exclusion, COG (top right), BADD (bottom left), and GkCOG (bottom right)

extremes. Fig. 2 (bottom left) shows the results for BADD. For large s (MOM), BADD yields 0.4 for $s < 1$, and jumps to $(0.4 + 0.9)/2 = 0.65$ for $s = 1$. For small s, BADD yields about 0.46, and jumps to 0.4 for $s = 0$. GkCOG (Fig. 2, bottom right) shows a very similar behavior, except for the boundary 0.5 instead of 0.46, and it does not show the jump for $s = 0$.

In our third experiment we consider a triangle with peak 0.5, width 1, height 1, and an exclusion at the position 0.6, i.e. the height at 0.6 is varied between $0, 0.1, \ldots, 1$ (Fig. 3, top left). For the different height levels, COG (Fig. 3, top right) almost linearly interpolates between about 0.48 and about 0.5. BADD (Fig. 3, bottom left) interpolates between two extremes: a concave curve from about 0.48 ($s = 0$) to 0.55 ($s = 1$), and a constant 0.5. Similar as in the previous example, BADD exhibits an edge from 0.49 ($s = 0$) to 0.5 ($s = 0.1$). GkCOG (Fig. 3, bottom right) is very similar but without the edge.

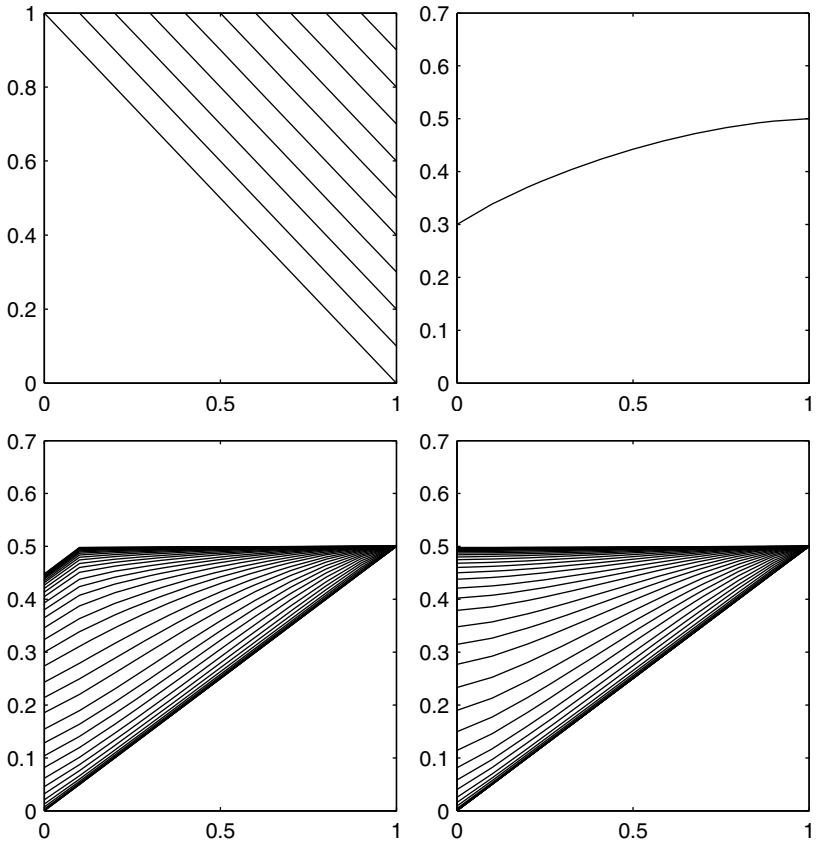


Fig. 4 Experiment 4: truncated trapezoids, COG (top right), BADD (bottom left), and GkCOG (bottom right)

In our fourth experiment we consider a truncated trapezoid with a right slope of 1 at the edge positions $0, 0.1, \ldots, 1$ (Fig. 4, top left). COG (Fig. 4, top right) yields a convex interpolation from 0.3 to 0.5. BADD (Fig. 4, bottom left) interpolates between a slope from $(0,0)$ to $(1,0.5)$ and a constant 0.5 with an edge at $s = 0$. GkCOG (Fig. 4, bottom right) is again very similar except for the edge.

6 Conclusions

Defuzzification is an important and continuously very active field of research since the early 1990s. Defuzzification converts a fuzzy set to a crisp value or set. Standard defuzzification methods are *center of gravity (COG)* and *mean of maxima (MOM)*. Many alternative approaches to defuzzification have been proposed. Parametric defuzzification provides a unified approach to defuzzification. A popular parametric defuzzification method is based on the basic defuzzification distribution (BADD).

BADD contains COG as a special case and can also be used to approximate MOM. We have introduced a kernelized variant of COG and have shown that BADD can be interpreted as kernelized COG with polynomial kernels. We have also considered kernelized COG with the popular Gaussian kernel, which we call Gaussian-kernelized COG (GkCOG). Experiments with four representative cases illustrate that parametric defuzzification allows to realize different defuzzification behavior depending on the chosen parameter values. Moreover, the experiments show that BADD often exhibits unsmooth behavior (edges in the defuzzification characteristics), which are avoided when using the Gaussian kernel (GkCOG).

This paper has introduced the concept of kernelized defuzzification. It has specifically considered kernelization of COG with polynomial and Gaussian kernels and experiments with one-dimensional data. Future work will include the kernelization of other defuzzification methods (parametric and non-parametric, one-dimensional and multi-dimensional) and a more detailed study how the properties of the kernel function relate to the properties of the resulting kernelized defuzzification function.

References

[1] Banaiyan, A., Fakhraie, S.M., Mahdiani, H.R.: Cost-performance co-analysis in VLSI implementation of existing and new defuzzification methods. In: International Conference on Computational Intelligence for Modelling Control and Automation, International Conference on Intelligent Agents, Web Technologies and Internet Commerce, Vienna, Austria, pp. 828–833 (2005)

[2] Broekhoven, E.V., Baets, B.D.: Fast and accurate center of gravity defuzzification of fuzzy system outputs defined on trapezoidal fuzzy partitions. Fuzzy Sets and Systems 157(7), 904–918 (2006)

[3] Cordón, O., Herrera, F., Márquez, F.A., Peregrín, A.: A study on the evolutionary adaptive defuzzification methods in fuzzy modeling. International Journal of Hybrid Intelligent Systems 1(1), 36–48 (2004)

[4] Coupland, S.: Type-2 fuzzy sets: Geometric defuzzification and type-reduction. In: IEEE Symposium Series on Computational Intelligence, IEEE Symposium on Foundations of Computational Intelligence, Honolulu, Hawaii, USA, pp. 622–629 (2007)

[5] Coupland, S., John, R.: A fast geometric method for defuzzification of type-2 fuzzy sets. IEEE Transactions on Fuzzy Systems 16(4), 929–941 (2008)

[6] Filev, D.P., Yager, R.R.: A generalized defuzzification method via BAD distributions. International Journal of Intelligent Systems 6, 687–697 (1991)

[7] Genther, H., Runkler, T.A., Glesner, M.: Defuzzification based on fuzzy clustering. In: IEEE International Conference on Fuzzy Systems, Orlando, pp. 1645–1648 (1994)

[8] Greenfield, S., Chiclana, F., Coupland, S., John, R.: The collapsing method of defuzzification for discretised interval type-2 fuzzy sets. Information Science 179(13), 2055–2069 (2009)

[9] Greenfield, S., Chiclana, F., Coupland, S., John, R.: Type-2 defuzzification: Two contrasting approaches. In: IEEE International Conference on Fuzzy Systems, Barcelona, Spain, pp. 1–7 (2010)

[10] Halgamuge, S.K., Runkler, T.A., Glesner, M.: On the neural network based defuzzification. In: IEEE International Conference on Fuzzy Systems, New Orleans, pp. 463–469 (1996)

[11] Hathaway, R.J., Huband, J.M., Bezdek, J.C.: Kernelized non–Euclidean relational fuzzy c–means algorithm. In: IEEE International Conference on Fuzzy Systems, Reno, pp. 414–419 (2005)
[12] Hellendoorn, H., Thomas, C.: On defuzzification in fuzzy controllers. Tech. Rep. SICSL–92/3, SIEMENS AG, München (1992)
[13] Hernandez, A.A.M., Hernandez, F.A.M., Rubio, A.P.: A multi-objective evolutionary algorithm with an interpretability improvement mechanism for linguistic fuzzy systems with adaptive defuzzification. In: IEEE International Conference on Fuzzy Systems, Barcelona, Spain, pp. 1–7 (2010)
[14] Korytkowski, M., Nowicki, R., Scherer, R., Rutkowski, L.: MICOG defuzzification rough-neuro-fuzzy system ensemble. In: IEEE International Conference on Fuzzy Systems, Barcelona, Spain, pp. 1–6 (2010)
[15] Leekwijck, W.V., Kerre, E.E.: Continuity focused choice of maxima: Yet another defuzzification method. Fuzzy Sets and Systems 122(2), 303–314 (2001)
[16] Lees, M.J., Campbell, D.A., Devlin, J.C.: A high-speed reconfigurable defuzzification architecture for true centre-of-gravity computations. In: Australian New Zealand Conference on Intelligent Information Systems, Adelaide, pp. 224–227 (1996)
[17] Linda, O., Manic, M.: Importance sampling based defuzzification for general type-2 fuzzy sets. In: IEEE International Conference on Fuzzy Systems, Barcelona, Spain, pp. 1–7 (2010)
[18] Linda, O., Manic, M.: Centroid density of interval type-2 fuzzy sets: Comparing stochastic and deterministic defuzzification. In: IEEE International Conference on Fuzzy Systems, Taipei, Taiwan, pp. 1560–1567 (2011)
[19] Lizárraga, G., Sepúlveda, R., Montiel, O., Castillo, O.: Modeling and simulation of the defuzzification stage using XILINX system generator and simulink. In: Castillo, O., Melin, P., Kacprzyk, J., Pedrycz, W. (eds.) Soft Computing for Hybrid Intelligent Systems. SCI, vol. 154, pp. 333–343. Springer, Heidelberg (2008)
[20] Mahdiani, H.R., Banaiyan, A., Fakhraie, S.M.: Hardware implementation and comparison of new defuzzification techniques in fuzzy processors. In: International Symposium on Circuits and Systems, Island of Kos, Greece (2006)
[21] Maity, K., Maiti, M.: Possibility and necessity constraints and their defuzzification — a multi-item production-inventory scenario via optimal control theory. European Journal of Operational Research 177(2), 882–896 (2007)
[22] Mercer, J.: Functions of positive and negative type and their connection with the theory of integral equations. Philosophical Transactions of the Royal Society A 209, 415–446 (1909)
[23] Opricovic, S., Tzeng, G.H.: Defuzzification within a multicriteria decision model. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 11(5), 635–652 (2003)
[24] Palm, R., Driankov, D.: Fuzzy inputs. Fuzzy Sets and Systems 70(2), 315–335 (1995)
[25] Patel, A.V., Mohan, B.M.: Some numerical aspects of center of area defuzzification method. Fuzzy Sets and Systems 132(3), 401–409 (2002)
[26] Pfluger, N., Yen, J., Langari, R.: A defuzzification strategy for a fuzzy logic controller employing prohibitive information in command formulation. In: IEEE International Conference on Fuzzy Systems, San Diego, pp. 717–723 (1992)
[27] Rey, A., Varela, B.A., Martínez, A.C.: Study on various defuzzification methods for fuzzy clustering algorithms to improve ROIs detection in lung CTs. In: IEEE International Conference on Fuzzy Systems, Taipei, Taiwan, pp. 2123–2130 (2011)

[28] Roychowdhury, S., Pedrycz, W.: A survey of defuzzification strategies. International Journal of Intelligent Systems 16(6), 679–695 (2001)
[29] Runkler, T.A.: Selection of appropriate defuzzification methods using application specific properties. IEEE Transactions on Fuzzy Systems 5(1), 72–79 (1997)
[30] Runkler, T.A.: Kernelized non–Euclidean relational possibilistic c–means clustering. In: IEEE Three Rivers Workshop on Soft Computing in Industrial Applications, Passau (2007)
[31] Runkler, T.A., Glesner, M.: Defuzzification with improved static and dynamic behavior: Extended center of area. In: European Congress on Intelligent Techniques and Soft Computing, Aachen, pp. 845–851 (1993)
[32] Runkler, T.A., Glesner, M.: DECADE — Fast centroid approximation defuzzification for real time fuzzy control applications. In: ACM Symposium on Applied Computing (SAC 1994), pp. 161–165. ACM Press, Phoenix (1994)
[33] Runkler, T.A., Glesner, M.: Defuzzification and ranking in the context of membership value semantics, rule modality, and measurement theory. In: European Congress on Intelligent Techniques and Soft Computing, Aachen, pp. 1206–1210 (1994)
[34] Runkler, T.A., Glesner, M.: Defuzzification as crisp decision under fuzzy constraints – new aspects of theory and improved defuzzification algorithms. In: Kruse, R., Gebhardt, J., Palm, R. (eds.) Fuzzy Systems in Computer Science, pp. 251–260. Vieweg, Wiesbaden (1994)
[35] Runkler, T.A., Glesner, M.: Multidimensional defuzzification — fast algorithms for the determination of crisp characteristic subsets. In: ACM Symposium on Applied Computing (SAC 1995). ACM Press, Nashville (1995)
[36] Saade, J.J., Diab, H.B.: Defuzzification techniques for fuzzy controllers. IEEE Transactions on Systems, Man, and Cybernetics, Part B 30(1), 223–229 (2000)
[37] Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT Press, Cambridge (2002)
[38] Sepúlveda, R., Montiel, O., Lizárraga, G., Castillo, O.: Modeling and Simulation of the Defuzzification Stage of a Type-2 Fuzzy Controller Using the Xilinx System Generator and Simulink. In: Castillo, O., Pedrycz, W., Kacprzyk, J. (eds.) Evolutionary Design of Intelligent Systems in Modeling, Simulation and Control. SCI, vol. 257, pp. 309–325. Springer, Heidelberg (2009)
[39] Sepúlveda, R., Montiel, O., Castillo, O., Melin, P.: Modelling and simulation of the defuzzification stage of a type-2 fuzzy controller using VHDL code. Control and Intelligent Systems 39(1) (2011)
[40] Sin, S.K., de Figueiredo, R.J.P.: Fuzzy system design through fuzzy clustering and optimal predefuzzification. In: IEEE International Conference on Fuzzy Systems, San Francisco, pp. 190–195 (1993)
[41] Sladoje, N., Lindblad, J., Nyström, I.: Defuzzification of spatial fuzzy sets by feature distance minimization. Image Vision Computing 29(2-3), 127–141 (2011)
[42] Starczewski, J.T.: On Defuzzification of Interval Type-2 Fuzzy Sets. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 333–340. Springer, Heidelberg (2008)
[43] Wang, W.J., Luoh, L.: Simple computation for the defuzzifications of center of sum and center of gravity. Journal of Intelligent and Fuzzy Systems 9(1-2), 53–59 (2000)
[44] Yager, R.R., Filev, D.P.: Defuzzification under constraints and forbidden zones. Tech. Rep. MII–1214, Machine Intelligence Institute, Iona College (1991)
[45] Yager, R.R., Filev, D.P.: SLIDE: A simple adaptive defuzzification method. IEEE Transactions on Fuzzy Systems 1(1), 69–78 (1993)
[46] Zadeh, L.A.: Fuzzy sets. Information and Control 8, 338–353 (1965)

Part II
Hybrid Intelligent Systems

The Algorithm Selection Problem on the Continuous Optimization Domain*

Mario A. Muñoz, Michael Kirley, and Saman K. Halgamuge

Abstract. The problem of algorithm selection, that is identifying the most efficient algorithm for a given computational task, is non-trivial. Meta-learning techniques have been used successfully for this problem in particular domains, including pattern recognition and constraint satisfaction. However, there has been a paucity of studies focused specifically on algorithm selection for continuous optimization problems. This may be attributed to some extent to the difficulties associated with quantifying problem "hardness" in terms of the underlying cost function. In this paper, we provide a survey of the related literature in the continuous optimization domain. We discuss alternative approaches for landscape analysis, algorithm modeling and portfolio development. Finally, we propose a meta-learning framework for the algorithm selection problem in the continuous optimization domain.

1 Introduction

A continuous optimization problem is such that, given a function $f : \mathbb{R}^n \mapsto \mathbb{R}$, we want to find $\mathbf{x}^\star = \arg\min f(\mathbf{x})$. When solved in a computer, a *search algorithm* samples from the very large but finite *search set*, $\mathcal{X} \subset \mathbb{R}^n$. Each observation $\mathbf{x}_i \in \mathcal{X}$ has an associated output value $y_i \in \mathcal{Y}$ such that $y_i \approx f(\mathbf{x}_i)$, where $\mathcal{Y} \subset \mathbb{R}$ is the *objective set*. The algorithm aims to find one or more candidate solutions $\mathbf{x}_o \in \mathcal{X}, y_o \approx f(\mathbf{x}_o)$, such that $|y_o - y^\star| \ll \delta$, where $y^\star = f(\mathbf{x}^\star)$ and $\delta \to 0$. It is expected that the algorithm produces a solution of acceptable quality after a bounded number of function evaluations. The opposite is known as *premature convergence*.

Mario A. Muñoz · Saman K. Halgamuge
Department of Mechanical Engineering,
The University of Melbourne, Parkville, Victoria, 3010, Australia
e-mail: mariom@student.unimelb.edu.au, saman@unimelb.edu.au

Michael Kirley
Department of Computing and Information Systems, The University of Melbourne,
Parkville, Victoria, 3010, Australia
e-mail: mkirley@unimelb.edu.au

* This work has been partially funded through a 2012-2013 DAAD/Go8 Grant.

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 75–89.
springerlink.com

Premature convergence is related to the nature of the search algorithm, as each algorithm exploits differently the information obtained by sampling f. Therefore, unless some restrictions are in place, it is optimistic to expect that an algorithm would work well across a wide range of functions [78]. Due to the plethora of available algorithms, it is non-trivial to know which one is able to exploit the information more efficiently [29]. This is an instance of the well known algorithm selection problem. In this paper we propose a framework based on meta-learning for the algorithm selection problem. For this purpose, we review the literature about the different stages of the new framework — namely landscape analysis, meta-learning models and algorithm portfolios. Then, we outline the requirements for implementation of the new framework.

The paper is organized as follow: Section 2 presents the algorithm selection problem for continuous optimization, and the related parameter tuning problem. Section 3 describes the characteristics that make an optimization problem difficult and it reviews different methods for landscape analysis. Section 4 discusses how machine learning techniques have been employed to solve the algorithm and parameter selection problems. Section 5 analyzes the related works in algorithm portfolio design. Section 6 presents our meta-learning based framework for the algorithm selection problem. Finally, Section 7 discusses avenues for further research.

2 Algorithm Selection

Rice [56] defined the algorithm selection framework as a loose methodology that relates problems and solution methods through performance and problem characteristics. This framework did not provide specific methods for implementation, which is one of the reasons it has not been thoroughly explored. However, in the last decades, meta-learning has been favored as implementation method with demonstrated success in different problem domains [62]. Meta-learning exploits data obtained from previous experiments by constructing models that can be used for prediction, using machine learning techniques [28]. Figure 1 presents a summary of this implementation adapted to continuous optimization problems. In this figure, $\mathcal{F}$ is the very large, amorphous, high dimensional and hard to define *function set*, for which $f \in \mathcal{F}$. Let $\mathcal{A}$ be the large and diverse *algorithm set*, and $a \in \mathcal{A}$ be one of the many algorithms capable of searching for $\mathbf{x}_o$ in $\mathcal{X}$. The cost of running a in f can be measured by a function $\rho(f,a)$. Let $\mathcal{P} \subset \mathbb{R}$ be the set of feasible values of $\rho(f,a)$, called the *performance set*. Then, the algorithm selection problem is to find $a_o = \arg\min \rho(f,a)$ with f constant. It is noteworthy to point out that this problem cannot be solved directly. Hence, let $\mathcal{C} \subset \mathbb{R}^m$ be the *set of function characteristics*. This set includes known attributes of f such as the dimension, but also measurements about the occurrence of certain structures known to pose difficulties for a [57, 73]. Characteristics are important as they provide some order and coherence to the complicated problem space by imposing a lower dimensional coordinate system [57]. Characteristics can be calculated through user defined functions known as *landscape analysis* methods, $c(\mathbf{x},y)$. These functions should be designed such that varying complexities are

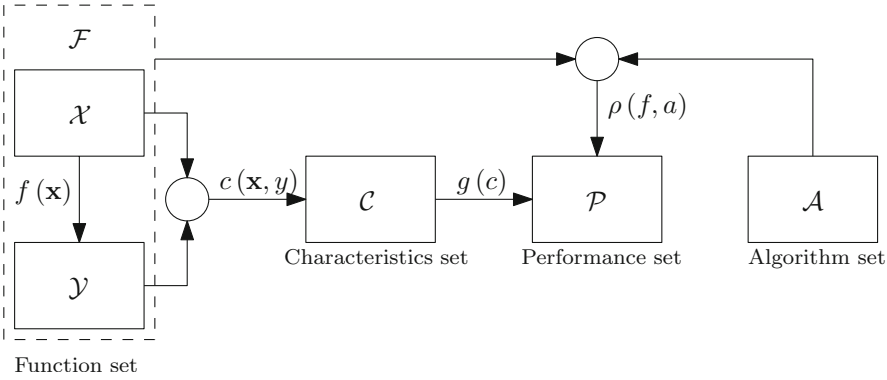


Fig. 1 Summary of the algorithm selection framework for the continuous optimization domain

exposed, structural properties are captured, and advantages and limitations of the different algorithms are related to them.

Let $g : \mathcal{C} \mapsto \mathcal{P}$ be a function that forecasts the performance based on the characteristics. Consider the existence of a subset of functions from $\mathcal{F}$ such that we know the values of $c(\mathbf{x}, y)$ and $\rho(f, a)$ for all $a \in \mathcal{A}$. Then, it is possible to use a machine learning technique to identify the function g. These *empirical performance models* provide a way to forecast the performance of an algorithm when a new problem is presented. The whole process can be automated if the results of several models are compared through an objective procedure.

The algorithm selection framework does not consider the algorithm parameter, θ, which controls the way that the search is carried out. This parameter can potentially adapt a to f if it is properly tuned, and it can appreciably change the overall performance [8]. This implies that an optimal θ for one function might not be appropriate for others [35, 50]. Choosing θ for a given a is a time-consuming and non-trivial task, and considerable effort has gone into developing methods for parameter selection that can be categorized as *parameter tuning* and *parameter control* [16]. Tuning keeps the parameters constant during the run, while control modifies them. Both approaches have advantages and disadvantages that have been thoroughly discussed in the literature [16, 35, 50].

Meta-learning is compatible with both parameter tuning and control [62]. If two instances of the same algorithm differ only in one parameter, we can consider them as two completely different algorithms [57]. This approach was followed by Hutter *et al.* [31, 30, 32] for tuning randomized algorithms in the context of boolean satisfiability problems, and by Muñoz *et al.* [45] for tuning the Covariance Matrix Adapted Evolutionary Strategy. Therefore, parameter selection can be seen as a component of the algorithm selection problem. As such, assume that g is not only dependent of the landscape characteristics c but also from θ. In fact, if we assume that c and θ are representations of f and a respectively, then $\rho(f, a) \equiv g(c, \theta)$.

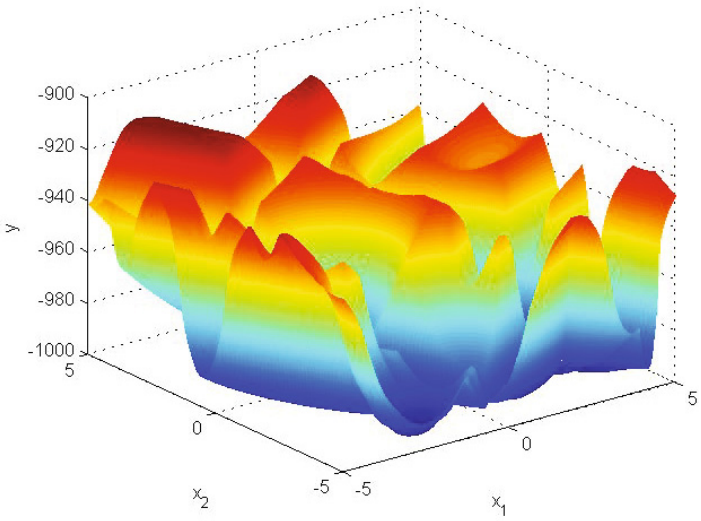
Even though meta-learning provides a clear avenue for implementation, every problem domain has specific issues to be considered, e.g. characteristic quantification, methods for selecting algorithms, and issues with uncertainties. One of the most important issues is the characteristic quantification. It is important to have a good understanding of what makes a continuous optimization difficult. For that purpose, in the next section we will discuss the *search landscape* metaphor and different methods used in landscape analysis of optimization problems.

3 Landscape Analysis

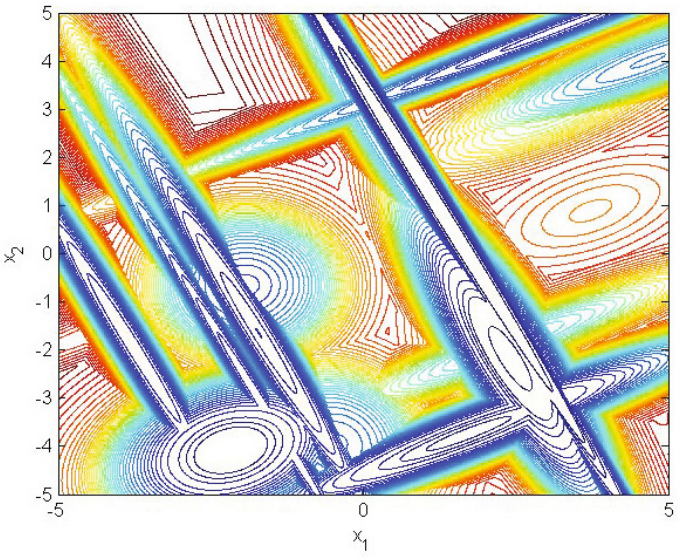
To understand what makes an optimization problem difficult, we employ the search landscape metaphor. Consider a surface in a three dimensional space composed of ridges, valleys and basins, such as the ones shown in Fig. 2. In this surface the highest or deepest areas represent the optimal points. The objective of the search is to navigate the surface until such areas are found. This metaphor helps us to understand what is needed for a successful search [53], and allows us to describe features in the landscape that are influential, even though most practical problems will have a dimension several orders of magnitude larger than two. These features — which are defined on detail in [44] — can be described qualitatively in cases where the knowledge about the function is complete. However, in cases where the only information available are the pairs $(\mathbf{x}_i, y_i)$ these attributes are usually unknown. Therefore, a *landscape analysis* technique is used to provide a measure that quantifies one or several attributes.

A number of landscape analysis methods have appeared in the past two decades. Table 1 presents a summary of some well-known landscape analysis methods and their underlying concepts. The measures have been classified into two groups: global and local. The former takes the whole sample to produce the measure while the later calculates the average of evaluating a condition over each observation and its neighborhood. Global measures have the advantage that samples extracted during an experiment can be reused to calculate different measures. However, they do not provide details about the locality of the landscape. Nevertheless, local measures can become intractable when the sample is too large, as each observation has to be analyzed independently. Other disadvantage of local measures is that samples obtained in one type of experiment are not reusable, e.g. time series measures require that a random walk experiment, while the basin of attraction measures require a local search experiment. Samples could be reusable if an intermediate processing step is placed. In previous work [44] we have demonstrated a procedure to calculate local measures from scattered data —extracted using random sampling— for two dimensional problems. However, this approach is not scalable. This is due to the sensitivity that local measures have to the neighborhood definition.

Other authors have identified limitations in the landscape analysis methods. Their application requires a sufficient number of observations, which grows exponentially as the dimension of the search space increases. This establishes a difference between



(a)



(b)

Fig. 2 Landscape for Gallagher's Gaussian 21-hi Peaks function in two dimensions from the Comparing Continuous Optimization Benchmark. Figure (a) show a three dimensional rendering of the function, while Fig. (b) show a contour plot. This function is multimodal, without global structure, non-separable, homogeneous with medium sized anisotropic basins of attractions.

Table 1 Summary of some well-known landscape analysis methods. Local measures calculate the average of evaluating a condition over each observation and its neighborhood, while global measures use the whole dataset to produce the measure.

Type	Concept	Measure	References
Local	Time Series	Kolmogrov complexity	[9]
		Correlation length	[64]
		Information content of random walk	[71, 72]
		Random walk correlation function	[76, 77]
	Fitness clouds	Negative slope coefficient	[68, 69, 67, 70]
	Evolvability	Fitness distributions	[7]
		Locality	[21]
		Fitness evolvability portraits	[61]
	Landmarking	Basin of attraction distributions	[22, 12, 17, 18, 54]
	Markov Models	Basin of attraction estimations	[2]
	Others	Phase transitions	[1]
		Ruggedness coefficient	[3]
		Information Landscapes	[8]
		Path diversity	[10]
		Motif difficulty	[38]
		Fourier transformations	[59]
Global	Linear correlation	Fitness distance correlation	[34]
		Multiple correlation coefficient	[41]
	Epistasis	Epistasis variance	[15]
		Bit-wise epistasis	[19]
		Walsh Transformation	[27]
		Bit decidability	[48]
		Analysis of variance tables	[55]
		Epistasis correlation	[58]
		Entropic epistasis	[60]
	Other	Dispersion	[39]

theoretical results and empirical estimators, whose precision changes as the number of observations increases to infinity [33]. Hence, a large amount of computation has to be made to obtain precise estimators and theoretically they cannot be calculated in polynomial time [26, 46, 66]. This also explains why statistical analysis can be artificially "fooled" by giving a special weight to insufficiently sampled regions of the landscape [66]. However, statistical measures are by nature approximate. The real question is how much information is actually lost and if its possible to deal with such losses. Also, providing a single global measure to analyze a whole landscape is overly optimistic and several measures may be necessary [6, 46, 61]. There is, without a doubt, another form of the no free lunch theorem [78] at work in this situation.

Our interest is to obtain as much information possible for a reasonable expenditure of effort. However, care must be taken into selecting methods that provide co-linear measurements [75].

Note that even if the analysis is performed, the results could only be applicable to the current representation [40]. To solve this issue, it has been suggested to use the Metropolis algorithm to get an initial sample and extract the measures while the optimization algorithm is running [47]. However, the bias imposed by the search algorithm can produce deceiving results [44]. It is obvious that difficulty of a problem can only be measured relative to the algorithm used to solve it [27]. Hence, it is necessary to relate the landscape features to the search cost. Otherwise, the resulting measures fail to account for much, if any, of the variability on problem difficulty [75]. However, meta-learning provides an avenue to solve various of these difficulties.

We will continue the discussion on how landscape analysis fits into the general framework in Sec. 6. Also, we can find examples on how landscape analysis has been applied to create meta-learning models. The next section reviews this research area.

4 Meta-learning Models for Optimization Algorithms

As we mentioned in Sec. 2, meta-learning uses data obtained from previous experiments by constructing prediction models of the algorithm using a machine learning technique. Table 2 presents a summary related of works to meta-learning in the continuous optimization domain. Unlike *fitness prediction* [11] — where a model of the function is created, so only promising observations are actually evaluated — machine learning is used as a mean to identify relationships among functions and algorithms with the purpose of selecting algorithms, tuning parameters, or simply understanding the algorithm behavior.

Francois and Lavergne [20] suggested that statistical analysis, in particular regression, could be useful to identify trends in the algorithm behavior. Their experiments concluded that performance is a random variable that follows a gamma distribution. This affirmation was confirmed by Yeguas *et al.* [79]. Although in [20] it is proposed to relate algorithm classes to performance, there is not a specific methodology in how to determine such classes. This means that for each problem a new model has to be trained. Hence, the resulting models could not be realistically used for parameter tuning. A similar conclusion can be drawn of the works by Bartz-Beielstein *et al.* [4, 5].

Leyton-Brown *et al.* [36, 37] are one of the first to focus on the algorithm selection in the optimization domain. Their work using combinatorial problems demonstrated the practical application of meta-learning, and how it can be successful in actual applications. However, only deterministic algorithms were studied at this stage. The work of Hutter *et al.* [31, 30, 32] demonstrated that randomized algorithms also can be modeled following this approach. These works provide justification to the exploration of meta-learning into the continuous optimization domain.

Table 2 Summary of the application of meta-learning concepts in the optimization domain. It is noteworthy the paucity of works dealing with algorithm selection using meta-learning concepts for continuous optimization.

Problem	Application	Model type	Reference
Algorithm Selection	Scheduling	Bayesian Classifier	[13, 14]
		Linear regression	[42]
	Program induction	Linear regression	[24]
	Boolean satisfiability	Standard ridge regression	[31, 30]
		Random Forest	[32]
	Combinatorial auctions	Linear regression	[36, 37]
		Multivariate Adaptive Regression Splines	[36]
Parameter Selection	Continuous optimization	Generalized Linear Models	[20]
		Regression tree	[4, 5]
		Linear models	[79]
		Neural networks	[45]
	Program induction	Linear models	[24]
	Boolean satisfiability	Standard ridge regression	[30]

A set of models would allow the user to maintain an empirical database of problem-algorithm relationships in a compact format. The database would be useful to select a single algorithm to run or a group of algorithms that can be run sequentially or concurrently, with or without communication between each other. This type of collection is known as *algorithm portfolio* [23], which we will discuss in the following section.

5 Algorithm Portfolios

The concept behind algorithm portfolios is simple [51]: "Instead of betting the entire time budget in a single algorithm, how do we invest it in multiple algorithms?" In other words, a portfolio aims to improve on the performance of the component algorithms, in terms of expected computational cost and overall risk [23]. This concept has been explored for more than ten years [23], and it is closely related to the developments in *memetic algorithms* [43, 49], *hyper-heuristics* [25] and *hybrid algorithms* [73]. In general, a portfolio contains besides the algorithm set, a procedure called selector, whose purpose is to decide which a is the best for a given f [25, 28, 51, 52, 65, 73]. In some cases, the portfolio provides provides communication among algorithms through a *migration scheme* [51]. The portfolio approach has demonstrated computational advantages over individual algorithms particularly when high-variance methods are combined [23].

The performance of a portfolio depends of both its composing algorithms and the selector [49]. In fact, some portfolios are strictly preferable than others, as they provide a lower risk and also a lower expected computational cost. However, in some cases, these are conflicting objectives [23]. Hence, it is preferable to select algorithms that are mutually complementary so a synergy can develop between them [74]. Therefore, it is crucial to understand the relative strengths and weaknesses of the different algorithms in the portfolio for effective selection [63]. It has been suggested that meta-learning systems infer which and why certain algorithms work for specific classes of functions. As such, the information gained through meta-learning allows to systematize the insights to combine algorithms purposefully and even provide clues for new algorithm designs [28].

A final element to consider in portfolio development is the random nature of the performance measure $\rho(f,a)$. Since it is possible to have large variations of performance over different instances of the same problem [25], the overall performance is quite sensitive to the runtime distributions of the algorithms involved [23]. Fortunately, for many randomized algorithms such distributions closely resemble standard parametric distributions [31], usually gamma as discussed in Sec. 4. Hence, they can be described by certain sufficient statistics. By forecasting such statistics, a prediction of the entire distribution for an unseen instance can be obtained [31].

6 An Extension of the Algorithm Selection Framework

So far we have discussed three main areas of research in the continuous optimization domain: landscape analysis, algorithm modeling and portfolio design. We have also pointed out how these areas are related to the algorithm selection problem and meta-learning. Now, we propose the framework shown in Fig 3, which connects these research areas together. This extended framework is composed by two feedback loops. The first of such loops is the *analysis loop*, which starts at the junction α where the pairs $(\mathbf{x}_i, y_i)$ are fed into the landscape analysis stage. At this stage, different analysis methods work in parallel to produce a vector of estimated characteristics denoted as $\hat{\mathbf{c}} \in \mathcal{C}$. As we pointed out in Sec. 3, global measures have useful computational advantages, particularly the possibility to reuse data from previous experiments. However, there are two important factors to take into account: The level of uncertainty associated with landscape analysis methods in general and the inherent bias of the samples that have been extracted during the run. For the former, a possible avenue is to consider confidence intervals instead of a single value as the result of the analysis. For the later, weighted resampling might provide correction over the bias. This solution was demonstrated in a previous work [44], with promising results.

The vector $\hat{\mathbf{c}}$ is the input for a set of models, each one of them represents an available algorithm. We do not favor any particular machine learning method to model the algorithms, although it is desirable to have a method that recognizes the uncertainty associated with the inputs, and provides a confidence interval for the output. The result is the vector of performance predictions $\hat{\boldsymbol{\rho}} \in \mathcal{P}^m$, where m is

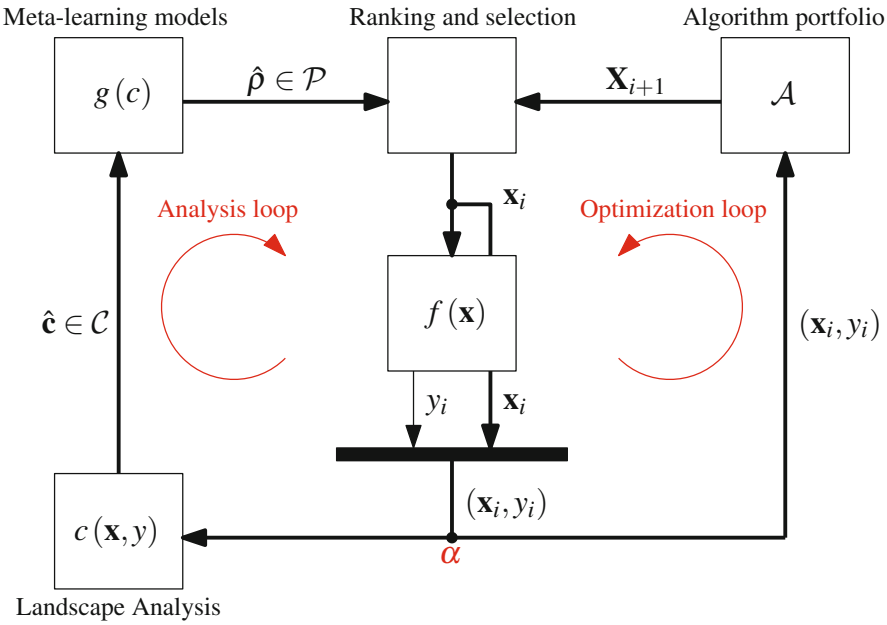


Fig. 3 Proposed extension of the algorithm selection framework. This new framework is composed of two loops: the optimization loop, where the search for promising solutions is carried out; and the analysis loop, where the selection for the best algorithm is made based on the landscape analysis.

the number of available algorithms. The predictions in this vector will be accurate depending on: the diversity in the knowledge base used to train the models, the relevance of the features, and the inherent randomness of the performance function. The results from this stage are used to create a ranking of the likelihood of each algorithm to create a new, potentially useful solution. At this point, it is important to consider the exploration/exploitation balance as well as the propagated uncertainty due to the landscape analysis and performance prediction.

The second loop is the *optimization loop*. It starts from the junction α, where the pairs $(\mathbf{x}_i, y_i)$ are fed into the algorithm portfolio. Besides producing new solutions to be evaluated, the portfolio shares information among constituent algorithms with the objective to improve the chances of producing useful solutions. For that purpose, it must be considered if the algorithms use a type of reinforcement learning, i.e. CMA-ES, or not, i.e. PSO. This is because data that is improperly supplied to the system might disrupt significantly the learning process. The resulting new solutions, $\mathbf{X}_{i+1}$, are transmitted to the ranking and selection mechanism, where the decision is taken into which one of them are fed into f.

In overall, the proposed framework provides different sources of information in order to produce a more extensive and detailed search. It also stores expertise that otherwise must be acquired by long, trial-and-error experiments. The framework

still provides flexibility into the selection of each component, such as which landscape analysis method to use, what type of model to implement, and importantly, which algorithms to select. It provides the opportunity of switching on and off algorithms depending on the case and to run concurrently as many algorithms as desired, unlike other portfolio approaches.

7 Discussion

In this paper we have discussed the relationship between landscape analysis, meta-learning models and algorithm portfolios to the algorithm selection framework as proposed by Rice [56]. We did so by reviewing the existing literature and proposing an extended framework that can be used for the algorithm selection in the continuous optimization domain. The proposed framework has several advantages: First, it enhances the search by providing additional sources of information that can be used to make decisions during the run. Second, it facilitates the storage of expertise that otherwise must be acquired by trial-and-error experiments. Third, it provides flexibility into the selection of each component, e.g. which landscape analysis method to use, what type of model to implement, and importantly, which algorithms to place in the portfolio. Finally, it provides the opportunity to run concurrently as many algorithms as desired, and being able to switch on and off those that are suitable at the time and place.

Our current work is focused in three areas of the framework. The first step is to develop a deeper understanding of some of the analysis methods in Tbl. 1. Our approach is to measure the uncertainty produced by the estimators through non-parametric statistical tests. The second step is to develop the meta-learning models. For this purpose, our approach is to use the confidence intervals of the landscape analysis as input to a machine learning strategy. The third step is to produce a ranking and selection mechanism. Our approach is to consider the uncertainty in the output models as part of the decision process. Algorithms with low uncertainty and high performance are deemed as the best choices, while algorithms with high uncertainty and low performance are deemed as the worst choices. The results so far are encouraging [44, 45].

References

[1] Achioptas, D., Naor, A., Peres, Y.: Rigorous location of phase transitions in hard optimization problems. Nature 435, 759–764 (2005)
[2] Anderson, E.: Markov chain modelling of the solution surface in local search. J. Oper. Res. Soc. 53(6), 630–636 (2002)
[3] Angel, E., Zissimopoulos, V.: On the hardness of the quadratic assignment problem with metaheuristics. Heuristics 8(4), 399–414 (2002)
[4] Bartz-Beielstein, T., Markon, S.: Tuning search algorithms for real-world applications: a regression tree based approach. In: CEC 2004, vol. 1, pp. 1111–1118 (2004)

[5] Bartz-Beielstein, T., Parsopoulos, K., Vrahatis, M.: Analysis of particle swarm optimization using computational statistics. In: ICNAAM 2004, pp. 34–37 (2004)
[6] Beck, J., Watson, J.: Adaptive search algorithms and Fitness-Distance correlation. In: Proc. 5th Metaheuristics Int. Conf. (2003)
[7] Borenstein, Y., Poli, R.: Fitness Distributions and GA Hardness. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 11–20. Springer, Heidelberg (2004)
[8] Borenstein, Y., Poli, R.: Information landscapes. In: GECCO 2005, pp. 1515–1522. ACM (2005)
[9] Borenstein, Y., Poli, R.: Kolmogorov complexity, optimization and hardness. In: CEC 2006, pp. 112–119 (2006)
[10] Boukeas, G., Halatsis, C., Zissimopoulos, V., Stamatopoulos, P.: Measures of Intrinsic Hardness for Constraint Satisfaction Problem Instances. In: Van Emde Boas, P., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2004. LNCS, vol. 2932, pp. 184–195. Springer, Heidelberg (2004)
[11] Branke, J., Schmidt, C.: Faster convergence by means of fitness estimation. Soft Comput. 9(1), 13–20 (2005)
[12] Brooks, C., Durfee, E.: Using Landscape Theory to Measure Learning Difficulty for Adaptive Agents. In: Alonso, E., Kudenko, D., Kazakov, D. (eds.) AAMAS 2000 and AAMAS 2002. LNCS (LNAI), vol. 2636, pp. 291–305. Springer, Heidelberg (2003)
[13] Carchrae, T., Beck, J.: Low knowledge algorithm control. In: AAAI 2004, pp. 49–54 (2004)
[14] Carchrae, T., Beck, J.: Applying machine learning to low-knowledge control of optimization algorihms. Comput. Intell. 21(4), 372–387 (2005)
[15] Davidor, Y.: Epistasis variance: A viewpoint on GA-hardness. In: Foundations of Genetic Algorithms. Morgan Kaufmann (1991)
[16] Eiben, A., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. 3(2), 124–141 (1999)
[17] Eremeev, A.V., Reeves, C.R.: Non-parametric Estimation of Properties of Combinatorial Landscapes. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R. (eds.) EvoIASP 2002, EvoWorkshops 2002, EvoSTIM 2002, EvoCOP 2002, and EvoPlan 2002. LNCS, vol. 2279, pp. 31–40. Springer, Heidelberg (2002)
[18] Eremeev, A.V., Reeves, C.R.: On Confidence Intervals for the Number of Local Optima. In: Raidl, G.R., Cagnoni, S., Cardalda, J.J.R., Corne, D.W., Gottlieb, J., Guillot, A., Hart, E., Johnson, C.G., Marchiori, E., Meyer, J.-A., Middendorf, M. (eds.) EvoIASP 2003, EvoWorkshops 2003, EvoSTIM 2003, EvoROB/EvoRobot 2003, EvoCOP 2003, EvoBIO 2003, and EvoMUSART 2003. LNCS, vol. 2611, pp. 224–235. Springer, Heidelberg (2003)
[19] Fonlupt, C., Robilliard, D., Preux, P.: A Bit-Wise Epistasis Measure for Binary Search Spaces. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 47–56. Springer, Heidelberg (1998)
[20] Francois, O., Lavergne, C.: Design of evolutionary algorithms-A statistical perspective. IEEE Trans. Evol. Comput. 5(2), 129–148 (2001)
[21] Galván-López, E., McDermott, J., O'Neill, M., Brabazon, A.: Defining locality as a problem difficulty measure in genetic programming. Genet. Program Evolvable Mach. 12(4), 365–401 (2011)
[22] Garnier, J., Kallel, L.: Efficiency of local search with multiple local optima. SIAM J. Discrete Math. 15(1), 122–141 (2002)

[23] Gomes, C., Selman, B.: Algorithm portfolios. Artif. Intell. 126(1-2), 43–62 (2001)
[24] Graff, M., Poli, R.: Practical performance models of algorithms in evolutionary program induction and other domains. Artif. Intell. 174, 1254–1276 (2010)
[25] Grobler, J., Engelbrecht, A., Kendall, G., Yadavalli, V.: Alternative hyper-heuristic strategies for multi-method global optimization. In: CEC 2010, pp. 1–8 (2010)
[26] He, J., Reeves, C., Witt, C., Yao, X.: A note on problem difficulty measures in black-box optimization: Classification, realizations and predictability. Evol. Comput. 15(4), 435–443 (2007)
[27] Heckendorn, R., Whitley, D.: Predicting epistasis from mathematical models. Evol. Comput. 7(1), 69–101 (1999)
[28] Hilario, M., Kalousis, A., Nguyen, P., Woznica, A.: A data mining ontology for algorithm selection and meta-mining. In: SoKD 2009, pp. 76–87 (2009)
[29] Hough, P., Williams, P.: Modern machine learning for automatic optimization algorithm selection. In: INFORMS AI/DM Workshop (2006)
[30] Hutter, F., Hamadi, Y., Hoos, H.H., Leyton-Brown, K.: Performance Prediction and Automated Tuning of Randomized and Parametric Algorithms. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 213–228. Springer, Heidelberg (2006)
[31] Hutter, F., Hamadi, Y., Hoos, H., Leyton-Brown, K.: Performance prediction and automated tuning of randomized and parametric algorithms: An initial investigation. In: The AAAI Workshop on Learning for Search: Schedule (2006)
[32] Hutter, F., Hoos, H., Leyton-Brown, K.: Tradeoffs in the empirical evaluation of competing algorithm designs. Tech. Rep. TR-2009-21, The University of British Columbia (2009)
[33] Jansen, T.: On classifications of fitness functions. Tech. Rep. CI-76/99, University of Dortmund (1999)
[34] Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: GECCO 1995, pp. 184–192. Morgan Kaufmann Publishers Inc. (1995)
[35] Jong, K.D.: Parameter setting in EAs: a 30 year perspective. In: Parameter Setting in Evolutionary Algorithms, Stud. Comput Intell., vol. 54, pp. 1–18. Springer (2005)
[36] Leyton-Brown, K., Nudelman, E., Shoham, Y.: Learning the Empirical Hardness of Optimization Problems: The Case of Combinatorial Auctions. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 556–572. Springer, Heidelberg (2002)
[37] Leyton-Brown, K., Nudelman, E., Shoham, Y.: Empirical hardness models: Methodology and a case study on combinatorial auctions. J. ACM 56(4), 22:1–22:52 (2009)
[38] Liu, L., Abbass, H., Green, D., Zhong, W.: Motif difficulty (MD): a predictive measure of problem difficulty for evolutionary algorithms using network motifs. Evol. Comput. 20(3), 321–347 (2012)
[39] Lunacek, M., Whitley, D.: The dispersion metric and the CMA evolution strategy. In: GECCO 2006, pp. 477–484. ACM, New York (2006)
[40] Merkuryeva, G., Bolshakovs, V.: Structural analysis of benchmarking fitness landscapes. Scientific Journal of Riga Technical University: Computer Sciences 42, 81–86 (2010)
[41] Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: GECCO 2011, pp. 829–836. ACM (2011)
[42] Messelis, T., Haspeslagh, S., Bilgin, B., Causmaecker, P.D., Berghe, G.V.: Towards prediction of algorithm performance in real world optimisation problems. In: BNAIC 2009, pp. 177–183 (2009)

[43] Molina, D., Herrera, F., Lozano, M.: Adaptive local search parameters for real-coded memetic algorithms. In: CEC 2005, vol. 1, pp. 888–895 (2005)
[44] Muñoz, M., Kirley, M., Halgamuge, S.: Landscape characterization of numerical optimization problems using biased scattered data. In: CEC 2012, pp. 2162–2169 (2012)
[45] Muñoz, M.A., Kirley, M., Halgamuge, S.K.: A Meta-learning Prediction Model of Algorithm Performance for Continuous Optimization Problems. In: Coello Coello, C.A., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 226–235. Springer, Heidelberg (2012)
[46] Müller, C.L., Sbalzarini, I.F.: Global Characterization of the CEC 2005 Fitness Landscapes Using Fitness-Distance Analysis. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcázar, A.I., Merelo, J.J., Neri, F., Preuss, M., Richter, H., Togelius, J., Yannakakis, G.N. (eds.) EvoApplications 2011, Part I. LNCS, vol. 6624, pp. 294–303. Springer, Heidelberg (2011)
[47] Naudts, B., Kallel, L.: A comparison of predictive measures of problem difficulty in evolutionary algorithms. IEEE Trans. Evol. Comput. 4(1), 1–15 (2000)
[48] Naudts, B., Suys, D., Verschoren, A.: Epistasis as a basic concept in formal landscape analysis. In: GECCO 1997, pp. 65–72. Morgan Kaufmann (1997)
[49] Özcan, E., Bilgin, B., Korkmaz, E.E.: Hill Climbers and Mutational Heuristics in Hyperheuristics. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 202–211. Springer, Heidelberg (2006)
[50] Pedersen, M.: Tuning & simplifying heuristical optimization. PhD thesis, University of Southampton (2009)
[51] Peng, F., Tang, K., Chen, G., Yao, X.: Population-Based algorithm portfolios for numerical optimization. IEEE Trans. Evol. Comput. 14(5), 782–800 (2010)
[52] Petrovic, S., Epstein, S., Wallace, R.: Learning a mixture of search heuristics. In: CP 2007 (2007)
[53] Reeves, C.: Fitness landscapes. In: Search Methodologies, pp. 587–610. Springer (2005)
[54] Reeves, C., Eremeev, A.: Statistical analysis of local search landscapes. J. Oper. Res. Soc. 55(7), 687–693 (2004)
[55] Reeves, C., Wright, C.: An experimental design perspective on genetic algorithms. In: Foundations of Genetic Algorithms, vol. 3, pp. 7–22. Morgan Kaufmann (1995)
[56] Rice, J.: The algorithm selection problem. In: Adv. Comput., vol. 15, pp. 65–118. Elsevier (1976)
[57] Rice, J.: Methodology for the algorithm selection problem. In: Proc. IFIP TC 2.5 Working Conference on Performance Evaluation of Numerical Software (1979)
[58] Rochet, S., Venturini, G., Slimane, M., El Kharoubi, E.M.: A Critical and Empirical Study of Epistasis Measures for Predicting GA Performances: A Summary. In: Hao, J.-K., Lutton, E., Ronald, E., Schoenauer, M., Snyers, D. (eds.) AE 1997. LNCS, vol. 1363, pp. 275–285. Springer, Heidelberg (1998)
[59] Rockmore, D., Kostelec, P., Hordijk, W., Stadler, P.: Fast fourier transform for fitness landscapes. Appl. Comput. Harmon Anal. 12(1), 57–76 (2002)
[60] Seo, D., Moon, B.: An Information-Theoretic analysis on the interactions of variables in combinatorial optimization problems. Evol. Comput. 15(2), 169–198 (2007)
[61] Smith, T., Husbands, P., Layzell, P., O'Shea, M.: Fitness landscapes and evolvability. Evol. Comput. 10(1), 1–34 (2002)
[62] Smith-Miles, K.: Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Comput. Surv. 41(1), 6:1–6:25 (2009)

[63] Smith-Miles, K., Hemert, J.v.: Discovering the suitability of optimisation algorithms by learning from evolved instances. Ann. Math. Artif. Intel. 61(2), 87–104 (2011)
[64] Stadler, P.: Landscapes and their correlation functions. J. Math. Chem. 20(1), 1–45 (1996)
[65] Streeter, M., Golovin, D., Smith, S.: Combining multiple heuristics online. In: AAAI 2007, vol. 2, pp. 1197–1203. AAAI Press (2007)
[66] Tomassini, M., Vanneschi, L., Collard, P., Clergue, M.: A study of fitness distance correlation as a difficulty measure in genetic programming. Evolutionary Computation 13(2), 213–239 (2005)
[67] Vanneschi, L.: Investigating problem hardness of real life applications. In: Genetic Programming Theory and Practice V, Genetic and Evolutionary Computation, pp. 107–124. Springer, US (2008)
[68] Vanneschi, L., Clergue, M., Collard, P., Tomassini, M., Vérel, S.: Fitness Clouds and Problem Hardness in Genetic Programming. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 690–701. Springer, Heidelberg (2004)
[69] Vanneschi, L., Tomassini, M., Collard, P., Vérel, S.: Negative Slope Coefficient: A Measure to Characterize Genetic Programming Fitness Landscapes. In: Collet, P., Tomassini, M., Ebner, M., Gustafson, S., Ekárt, A. (eds.) EuroGP 2006. LNCS, vol. 3905, pp. 178–189. Springer, Heidelberg (2006)
[70] Vanneschi, L., Valsecchi, A., Poli, R.: Limitations of the fitness-proportional negative slope coefficient as a difficulty measure. In: GECCO 2009, pp. 1877–1878. ACM, New York (2009)
[71] Vassilev, V., Fogarty, T., Miller, J.: Information characteristics and the structure of landscapes. Evol. Comput. 8(1), 31–60 (2000)
[72] Vassilev, V., Fogarty, T., Miller, J.: Smoothness, ruggedness and neutrality of fitness landscapes: from theory to application. In: Advances in Evolutionary Computing, pp. 3–44. Springer, New York (2003)
[73] Vassilevska, V., Williams, R., Woo, S.: Confronting hardness using a hybrid approach. In: Proc. 17th Ann. ACM-SIAM Symp. Discrete Algorithms, pp. 1–10. ACM, New York (2006)
[74] Vrugt, J., Robinson, B., Hyman, J.: Self-Adaptive multimethod search for global optimization in Real-Parameter spaces. IEEE Trans. Evol. Comput. 13(2), 243–259 (2009)
[75] Watson, J., Beck, J., Howe, A., Whitley, L.: Problem difficulty for tabu search in job-shop scheduling. Artif. Intell. 143(2), 189–217 (2003)
[76] Weinberger, E.: Correlated and uncorrelated fitness landscapes and how to tell the difference. Biol. Cybern. 63(5), 325–336 (1990)
[77] Weinberger, E., Stadler, P.: Why some fitness landscapes are fractal. J. Theor. Biol. 163(2), 255–275 (1993)
[78] Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1(1), 67–82 (1997)
[79] Yeguas, E., Joan-Arinyo, R., Luzón, M.V.: Modeling the performance of evolutionary algorithms on the root identification problem: A case study with PBIL and CHC algorithms. Evol. Comput. 19(1), 107–135 (2011)

Neuro-fuzzy Systems: A Short Historical Review

Detlef D. Nauck and Andreas Nürnberger

Abstract. When the popularity of fuzzy systems in the guise of fuzzy controllers began to rise in the beginning of the 1990s researchers became interested in supporting the development process by an automatic learning process. Just a few years earlier the backpropagation learning rule for multi-layer neural networks had been rediscovered and triggered a massive new interest in neural networks. The approach of combining fuzzy systems with neural networks into neuro-fuzzy systems therefore was an obvious choice for making fuzzy systems learn. In this chapter we briefly recall some milestones on the evolution of neuro-fuzzy systems.[1]

1 Introduction

The term *neuro-fuzzy systems* (also neuro-fuzzy methods or models) refers to combinations of techniques from neural networks and fuzzy systems [26, 50]. This typically does not mean that a neural network and a fuzzy system are used in some kind of combination, but that a fuzzy system is created from data by some kind of (heuristic) learning method that is motivated by learning procedures used in neural networks.

Neuro-fuzzy methods are usually applied, if a fuzzy system is required to solve a function approximation problem — or a special case of it like classification or

Detlef D. Nauck
BT, Research and Venturing, Ipswich, IP5 3RE, UK
e-mail: detlef.nauck@bt.com

Andreas Nürnberger
Faculty of Computer Science, Otto-von-Guericke University of Magdeburg,
39106 Magdeburg, Germany
e-mail: andreas.nuernberger@ovgu.de

[1] This paper is an updated and extended version of a conference paper of the authors that appeared at NAFIPS 2005 [49].

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 91–109.
springerlink.com

control [58] — and the otherwise manual design process should be supported or replaced by an automatic learning process. The (manual) design of a fuzzy system requires specification of fuzzy partitions (parameters) for each variable and a set of fuzzy rules (structure). If the fuzzy system does not perform well, structure or parameters or both must be modified accordingly. This can be a very lengthy and error-prone process that is effectively based on trial and error. In order to support this design process learning techniques based on sample data became a popular research topic at the beginning of the 1990s when fuzzy systems in the guise of fuzzy controllers first became successful and widely known.

The history of neuro-fuzzy systems can be roughly structured into first feed-forward systems for control and function approximation and later — mainly due to the success of fuzzy systems in control — approaches for classification and clustering problems, where interpretable solutions and the introduction of prior knowledge into the learning process is also quite often very beneficial. More recently, several researchers studied the usability of hierarchical and recurrent architectures. In the following, we discuss the major approaches that have been proposed for these fields, if possible, in chronological order.

2 Feed-Forward Architectures

One of the first works that proposed a combination of neural network learning methods with the concepts of fuzzy systems was proposed in 1985 by Keller and Hunt [29]. In this paper, the authors proposed an approach to stabilize the perceptron learning algorithm for classification problems using fuzzy techniques. They introduced a fuzzy membership of data items to the searched classes in order to improve the convergence of the learning algorithm. Motivated by this early work, several other approaches had been proposed that deal with the combination of neural networks and fuzzy systems and that have driven this field of research. In the area of approximate reasoning, for example, several approaches have been proposed in 1991 and 1992 [14, 28, 31, 30, 32, 57, 56]. These models are parts of fuzzy expert systems, or support fuzzy decision making with the help of neural networks. Since these methods do not integrate the neural network and fuzzy system structure in a homogenous architecture, but one adapts the parameters of the other in a cooperative way, we do not consider them as (hybrid) neuro-fuzzy system and thus do not cover them as part of this contribution. The same holds for approaches suggested by Miyoshi et al. and Yager and Filev in 1993 and 1992, respectively. In these approaches the fuzzy sets are not modified, but parametrized t-norms and t-conorms are used. Miyoshi et al. [39] proposed an approach to adapt parameters of these operators by backpropagation, while Yager and Filev suggest adaptive defuzzification strategies [77, 78]. Yager and Filev used a parametrized defuzzification operation and define a supervised learning algorithm to determine the parameters. However, even though we do not discuss co-operative approaches in detail, we will refer to them, if other (hybrid) approaches made use of the proposed more general

techniques and ideas. The same holds for problems of learnability and interpretability of (neuro) fuzzy systems, where some discussions can be found in [22, 64, 47, 11, 42]. For an overview that is focussed on neuro-fuzzy methods for rule generation see [38].

2.1 Control

Neuro-fuzzy controllers were together with neuro-fuzzy systems for function approximation the first neuro-fuzzy approaches. The principles and architectures are similar and the main difference is basically the learning mechanism. While function approximation models can use supervised learning based on a training set, controllers need to discover a model in a setting where target outputs are not known. Neuro-fuzzy controllers therefore use reinforcement learning and require either a model of or direct feedback from the process they are supposed to control.

2.1.1 ARIC and GARIC

On of the first neuro-fuzzy controllers was suggested by Berenji in 1992. The ARIC model (Approximate Reasoning-based Intelligent Control) implements a fuzzy controller by using several specialized feed-forward neural networks. The architecture of ARIC is similar to an adaptive critic, a special neural controller learning by reinforcement [75], and it generalizes the neural model of Barto et al. [3] to the domain of fuzzy control. ARIC consists of two neural modules, the ASN (Action Selection Network) and the AEN (Action state Evaluation Network). The AEN is an adaptive critic that is trained by backpropagation and that evaluates the actions of the ASN.

The ASN itself consists of two feed-forward three-layer neural networks. One network calculates a confidence value that is used to change the output of the second network which is a direct representation of a fuzzy controller. The input layer represents state variables of a process and the hidden units represent fuzzy rules. Their inputs are the antecedents, and their outputs the consequents of the rules. ARIC assumes that the rule base is known in advance. The output of the control network represents the defuzzified control value of the fuzzy controller. The learning algorithm modifies connections weights in the ASN and so indirectly the represented fuzzy sets.

Implementing learning by modifying connection weights was a popular approach in early neuro-fuzzy systems, but it was later shown that this leads to problems in interpreting the learning outcome [40].

The ARIC model was later extended to GARIC (Generalised ARIC, Fig. 1) [4, 5, 6]. Like ARIC it consists of an evaluation network (AEN) and an action network (ASN). The ASN does not use any weighted connections, but the learning process modifies parameters stored within the units of the network. The other network of the ASN which produces a confidence measure no longer exists.

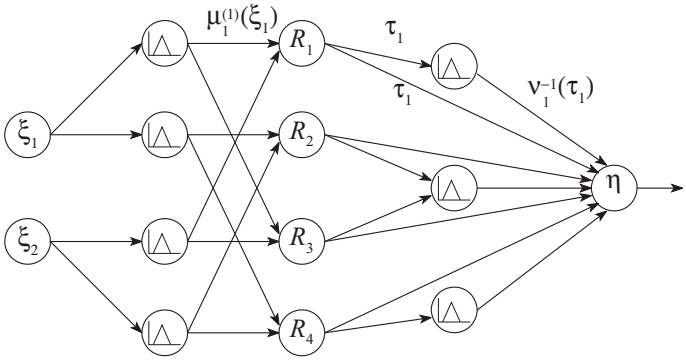


Fig. 1 GARIC represents a fuzzy system as a feed-forward network [5]

2.1.2 NEFCON

NEFCON is a model for neural fuzzy controllers proposed by Nauck in 1994 [44]. It is based on the architecture of the generic fuzzy perceptron [50] and implements a Mamdani-type fuzzy system. NEFCON is probably the first neuro-fuzzy system that tries to introduce the notion of interpretability by preventing identical linguistic terms being represented by more than one membership function, even though the general idea of a fuzzy perceptron has been proposed already earlier by Keller and Tahani as well as Pal and Mitra in 1992 [31, 54] and similar concepts have been discussed by Gupta and Rao in 1994 [18].

Like ARIC and GARIC, the learning algorithm for NEFCON is based on the idea of reinforcement learning but instead of an adaptive critic network it uses a fuzzy rule base to describe a fuzzy error.

Fig. 2 shows a NEFCON system with two input variables, one output variable and five rules. The unit R_3 for instance represents the rule

$$R_3\text{: } \textbf{If } \xi_1 \textbf{ is } \mu_2^{(1)} \textbf{ and } \xi_2 \textbf{ is } \mu_2^{(2)} \textbf{ then } \eta \textbf{ is } \nu_2.$$

The connections in NEFCON are weighted with fuzzy sets instead of real numbers, and some connections always have the same weight (illustrated by ellipses around connections) in order to ensure the integrity of the rule base.

Several learning methods have been proposed for this model. In [53] an overview is given. One major problem of all methods is, that they require at least some prior knowledge of the system to be controlled in order to define the (fuzzy) error signal that is used for learning.

Recently, models have been proposed that try to solve these fundamental problems using hierarchical models and Q-learning, see, e.g., [13, 15]. Q-learning has been already successfully used in combination with neural networks in order to control more complex systems, see e.g. [60].

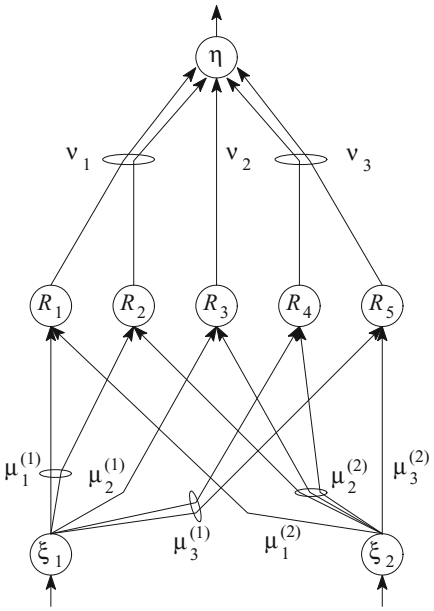


Fig. 2 A NEFCON system with two input variables and five rules

2.2 Approximation

Many neuro-fuzzy systems for function approximation are based on Takagi-Sugeno fuzzy systems, because they allow the application of gradient descent learning, if differentiable membership function (e.g. Gaussians) are used.

2.2.1 ANFIS

One of the first and still one of the popular neuro-fuzzy systems is Jang's ANFIS model proposed in 1991 [26, 23, 24, 25]. ANFIS (adaptive network-based fuzzy inference system) is a neuro-fuzzy method to determine the parameters of a Sugeno-type fuzzy model which is represented as a special feed-forward network (see Fig. 3). It encodes fuzzy rules of the form

$$R_r\text{: } \textbf{If } x_1 \textbf{ is } \mu_{j_1}^{(1)} \wedge \ldots \wedge x_n \textbf{ is } \mu_{j_n}^{(n)} \textbf{ then } y = \alpha_0^{(r)} + \alpha_1^{(r)} x_1 + \ldots + \alpha_n^{(r)} x_n.$$

Each node of the first layer is connected to exactly one of the n input variables and stores the three parameters of a membership function. The k nodes in the second layer represent the antecedents of the fuzzy rules. They compute the degree of fulfillment by multiplying the degrees of membership. The k nodes of the third layer compute the relative degree of fulfillment for each rule. The output values of the

rules are computed by the k nodes of layer 4. They store the consequent parameters. Each node in this layer is connected (not drawn in Fig. 3) to one node of layer 3 and to all input variables. The output node in layer 5 computes the overall output value. If the model must compute $m > 1$ output values, then there are m output nodes and mk nodes in layer 4.

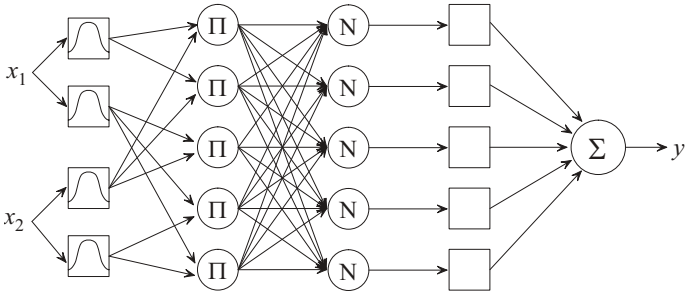


Fig. 3 ANFIS encodes a Sugeno-type fuzzy model in a feed-forward network structure [25]

ANFIS uses only differentiable functions, and therefore it is easy to apply standard gradient descent learning procedures from neural network theory. For ANFIS a mixture of backpropagation (BP) and least mean square estimation (LSE) is suggested by Jang [25]. BP is used to compute the updates of the antecedent parameters, i.e. the parameters of the fuzzy sets, and LSE is used to determine the updates for the consequent parameters, i.e. the coefficients of the linear combinations in the consequents.

ANFIS does not learn the structure of the fuzzy system, but it simply creates rules from all possible combinations of input fuzzy sets. Initial fuzzy partitions have to be specified. The consequent parameters are initialised by small random numbers.

2.2.2 Radial Basis Function Networks

Radial basis function networks (RBFN) are often connected to fuzzy systems, because the activation functions $h(||x-c||^2) = exp(-\frac{||x-c||^2}{2\sigma^2})$ of their hidden units can be interpreted as multidimensional membership functions. If this interpretation is assumed, then fuzzy rules can be extracted from an RBFN. To do this the RBF functions of the hidden units must be projected onto the individual dimensions. This way fuzzy sets are obtained that must be labelled with suitable linguistic terms. In general, this kind of rule generation suffers from the problem that the antecedent of the resulting fuzzy rule is not necessarily equivalent to the original corresponding RBF function. We only have equivalence if the area described by it is an axis-paralell hyperellipsoid and the product is used to compute the degree of fulfilment of the

extracted fuzzy rule. If min is used as a t-norm the support of a rule is equivalent to the smallest hyperbox which contains the hyperellipsoid described by $h(||x-c||^2) > \varepsilon$. If the RBFN uses a generalised radial basis function with an inverse co-variance matrix that is not diagonal it is no longer equivalent to a fuzzy system.

In 1993 Jang and Sun showed when a radial basis function network (RBFN) is equivalent to a TSK fuzzy model [27]. They found that the following conditions have to hold:

- The number of hidden units in the RBFN (receptive field units) is equal to the number of fuzzy if-then rules.
- The output of each fuzzy if-then rule is just a constant, i.e. the fuzzy system is a simplified special case of a TSK system which would normally have a linear combination as rule output.
- All membership functions in the fuzzy system are Gaussian functions with the same variance.
- The t-norm operator used to compute each rule's degree of fulfilment is the product.
- Both the RBFN and the fuzzy inference system use the same operation to compute the overall output, i.e.either weighted average or weighted sum.

Note that the third condition can be relaxed such that only the variances for each corresponding dimension (input variable) have to be identical. That means the RBFN can use basis function with an inverse covariance matrix that is a diagonal matrix and the elements on the diagonal need not be identical.

RBF networks have frequently be used to derive neuro-fuzzy approaches. For example, Fuzzy RuleNet [69] as discussed in Sect. 2.3.4 is an extension of the RuleNet model which is a special RBFN. The activation functions of the hidden units use the ∞-vector norm instead of the usual Euclidean vector norm. Thus the activation functions are defined over hyperboxes instead of hyperellipsoids [12, 70].

Fuzzy RuleNet is a typical approach that can be seen as being inspired by RBFN but having outgrown the limitation of functional equivalence to RBFN by using hyperboxes as the support of antecedents and max-min interference instead of product and weighted average. Similar hyperbox-oriented approaches have been presented by Berthold and Huber [10, 8, 9].

2.2.3 NEFPROX

NEFPROX [48] is like NEFCON based on a generic fuzzy perceptron and implements a Mamdani-type fuzzy system. Mamdani-type system are rarely used for function approximation purposes, because it is easier to train a Takagi-Sugeno-type system. NEFPROX has learning algorithms for structure learning and parameter learning, but it is typically less accurate than ANFIS.

2.3 Classification

Neuro-fuzzy classification systems became more popular in the second half of the 1990s. They are a special case of function approximators and their output is typically a fuzzy classification of a pattern, i.e. a vector of membership degrees that indicates membership to different classes. With the rising interest in data mining, fuzzy classifiers became more and more important in the fuzzy system community.

2.3.1 The NNDFR Model

A very interesting and atypical neuro-fuzzy model from 1991 is the NNDFR model (Neural Network Driven Fuzzy Reasoning) by Takagi and Hayashi [66] which was developed around the same time as ANFIS. NNDFR is based on common neural networks that are structured by fuzzy system techniques. Its main purpose is classification.

An NNDFR system is based on n input variables $x_1,\ldots,x_n$, an output variable y and k fuzzy rules $R_1,\ldots,R_k$. It consists of $k+1$ multi-layer feed-forward neural networks trained by backpropagation and representing the fuzzy rules (Fig. 4). The system cannot be used to extract the parameters of a fuzzy system from it. Strictly speaking, we would not consider it as a neuro-fuzzy system. However, the structure of the partial networks and the interpretation of the outputs are motivated by fuzzy system techniques.

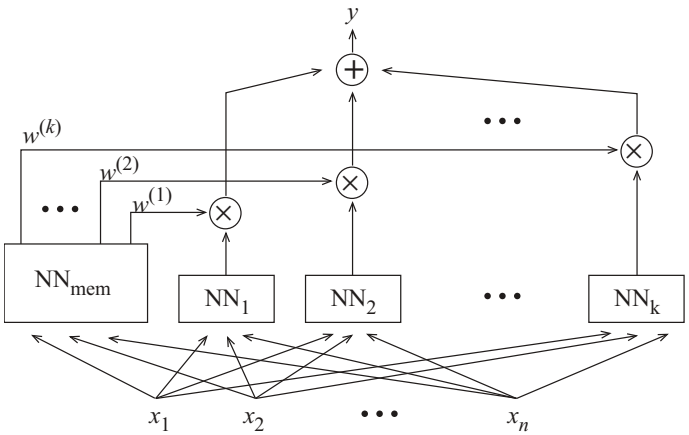


Fig. 4 Structure of an NNDFR system [66]

The linguistic rules used by the NNDFR model are of the form R_r: **If** $(x_1,\ldots,x_n)$ **is** A_r **then** $y = u_r(x_1,\ldots,x_n)$. This is not the usual form of linguistic rules used in fuzzy systems and is caused by the purely neural architecture. A_r is an n-dimensional

membership function. There is no combination of single membership values. In an NNDFR system the neural network $\mathrm{NN}_{\mathrm{mem}}$ provides for each rule R_r a value w_r that is interpreted as its degree of fulfillment.

The functions $u_1, \ldots, u_k$ are implemented by k neural networks $\mathrm{NN}_1, \ldots, \mathrm{NN}_k$ that determine the output values of the rules $R_1, \ldots, R_k$. The overall system output is given by $y = \sum_{r=1}^{k} w_r \, u_r(x_1, \ldots, x_n) / \sum_{r=1}^{k} w_r$.

2.3.2 FuNe-I

The neuro-fuzzy model FuNe-I [20, 21], proposed in 1992, is based on the architecture of a feed-forward neural network (Fig. 5). The network has five layers. The first layer contains a unit for each input variable and propagates the input values unchanged via weighted links to the second layer. This layer consists of units with sigmoid activation functions that are used to create membership functions. The third layer contains specialized units that are only used to represent fuzzy sets that do not touch the domain boundaries (see below). The units of the second and third layer propagate their activations via unweighted links to the fourth layer. Units from the second layer that have connections to the third layer are not connected to the fourth layer.

The fourth layer consists of units that represent fuzzy rules. Compared to other neuro-fuzzy approaches, the FuNe-I model is special because it uses three kinds of rules: the antecedents can be conjunctions or disjunctions, and there are rules with only one variable as antecedent (simple rules). A unit computes its activation — depending on the kind of rule it represents — by either a differentiable soft minimum, a differentiable soft maximum, or the identity function.

The fifth layer contains the output units that compute their input by a weighted sum and their activation by a sigmoid function. The FuNe-I model provides algorithms for structure and parameter learning and is one of the first neuro-fuzzy approaches that also considers rule learning.

FuNe-I was extended in 1994 to FuNe-II which can be used for fuzzy control problems. In a FuNe-II network a new output layer is created that is connected to the previous output layer. On the connections discrete samples of fuzzy sets are stored to represent control values. The activations of the new output units represent support points of a fuzzy set that must be defuzzified to obtain the final control value [19, 21].

2.3.3 NEFCLASS

NEFCLASS [45, 46, 41], proposed in 1995, is probably the first neuro-fuzzy approach that was able to handle missing values, both numeric and symbolic data in the same data set and to determine a rule-base fully automatically. NEFCLASS is also based on the idea of a generic fuzzy perceptron and focuses on creating small interpretable fuzzy rule bases.

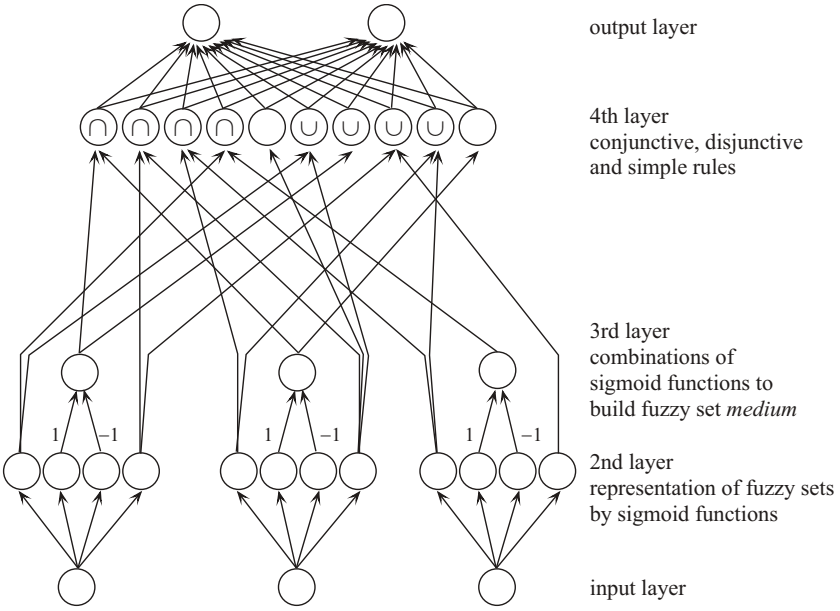


Fig. 5 The architecture of a FuNe-I system

The learning algorithm of NEFCLASS has two stages: structure learning and parameter learning. Rule (structure) learning is done by a variation of the approach by Wang and Mendel [73] which was extended to cover also symbolic patterns [43] and to use a rule performance measure for rule selection. In parameter learning the fuzzy sets are tuned by a simple backpropagation-like procedure that is based on a simple heuristics instead of a gradient descent approach. After learning NEFCLASS uses pruning strategies to reduce the number of rules as much as possible.

2.3.4 Fuzzy RuleNet

Fuzzy RuleNet [69], which was also proposed in 1995, is a neuro-fuzzy approach that is based on the structure of an radial basis function (RBF) network (see also Sect. 2.2.2). It is an extension of the RuleNet model, a special neural network, that can be seen as a variant of an RBF network [12, 70]. Instead of the usual radial basis functions — which represent hyperellipsoids — RuleNet uses hyperboxes for classification.

Fuzzy RuleNet allows hyperboxes to overlap. Each hyperbox represents a multidimensional fuzzy set given by a membership function in form of a hyperpyramid. By projecting the multidimensional fuzzy sets onto the individual dimensions we obtain triangular or trapezoidal fuzzy sets that describe the pattern features. The fuzzy classification rules obtained this way are equivalent to the multidimensional fuzzy sets, i.e. there is no loss of information as it would be in the case of hyperellipsoids used in fuzzy cluster analysis. The hyperboxes are created in a single cycle

through the data. The learning algorithm adjusts the sizes of hyperboxes by extending them to cover new data or shrinking them in case of conflicts. This way a rule base and the parameters (fuzzy sets) are created in a single loop.

If a Fuzzy RuleNet is used for classification it computes its output by a winner-takes-all procedure to find the class of a given input pattern. It is also possible to adjust the definition such that the outputs are computed by a weighted sum. This way Fuzzy RuleNet can be used for function approximation.

Similar approaches to Fuzzy RuleNet are sometimes called hyperbox-oriented rule learners, and were known as early as 1992 [62, 63]. Newer variations are also called fuzzy graphs [9, 7]. The idea is always to cover a set of data with hyperboxes and connect each hyperbox with an output value. Hyperbox-oriented fuzzy rule learning can create solutions for benchmark problems in pattern recognition or function approximation very fast. If there are no contradictions in the training patterns and if there is only one output variable, then hyperbox-oriented learning algorithms can create solutions with no errors on the training data. In the worst case this leads to a situation, where each training pattern is covered by its individual hyperbox.

3 Recurrent Systems

In contrast to pure feed-forward architectures that have a static input-output behavior, recurrent models are able to store information of the past, e.g. prior system states, and can be thus more appropriate for the analysis of dynamic systems (see, for example, discussions concerning the approximation and emulation capabilities of recurrent neural networks [59, 37, 74]). If pure feed-forward architectures are applied to these types of problems, e.g. prediction of time series data or physical systems, the obtained system data usually has to be preprocessed or restructured to map the dynamic information appropriately, e.g. by using a vector of prior system states as additional input. If we apply a fuzzy system, this may lead to an exponential increase of the parameters — if we want to cover the whole system state space — that soon becomes intractable.

Recurrent neuro-fuzzy systems (RNFSs) can be constructed in the same way as discussed above for feed-forward neuro-fuzzy systems. So, they are based either on a recurrent fuzzy system or a recurrent neural network structure. However, the design and the optimization of (hierarchical) recurrent systems is, due to the dynamics introduced by the feed back connections, more difficult than that of feed forward systems. In Fig. 6 an example of a hierarchical RNFS is shown.

Probably the first recurrent fuzzy system that was combined with a (neural network motivated) learning method was proposed by Gorrini and Bersini in 1994 [17]. The proposed system is a Sugeno-Takagi-like fuzzy system and uses fuzzy rules with a constant consequent. The internal variables of this system may be defined manually, if the designer has sufficient knowledge of the system that should be modeled. No learning method for the rule base itself was proposed except to initialize the rule base randomly. However, the authors propose a learning approach

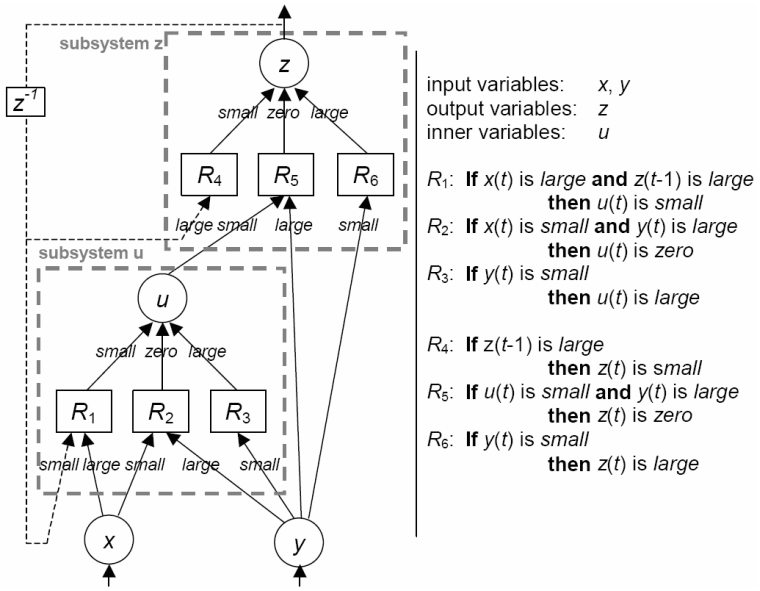


Fig. 6 Example of a simple hierarchical recurrent rule base consisting of two subsystems. The output of the system is reused by each subsystem as time-delayed input.

to optimize the parameters of a recurrent rule base, which was motivated by the real time recurrent learning algorithm [76]. According to Gorrini and Bersini the results for the approximation of a third order non-linear system for a given rule base was comparable to the approximation by a recurrent neural network. Unfortunately, a detailed discussion of the results was not given. Furthermore, the model had some insufficiencies. First of all, the structure has to be defined manually, since no learning methods for the rule base have been proposed. Furthermore, the learning is restricted to symmetric triangular fuzzy sets and the interpretability is not ensured, since the fuzzy sets are modified independently during learning. However, an extension to arbitrary (differentiable) fuzzy sets is easily possible.

Surprisingly, after this first model, for some time not much work had been published on recurrent systems that are also able to learn the rule base itself. Most likely the first models that were successfully applied to control — which, however, do not implement generic hierarchical recurrent models as described above — were proposed by Theocharis and Vachtsevanos in 1996 [68], Zhang and Morris in 1999 [80] and Lee and Teng in 2000 [35]. For example, Lee and Teng proposed a fuzzy neural network, which implements a modified Sugeno-Takagi-like fuzzy system with Gaussian-like membership functions in the antecedents and constant consequents. However, this model did not implement a fully recurrent system as shown in Fig. 6, but they restricted themselves to integrate feed back connections in the membership layer as depicted in Fig. 7.

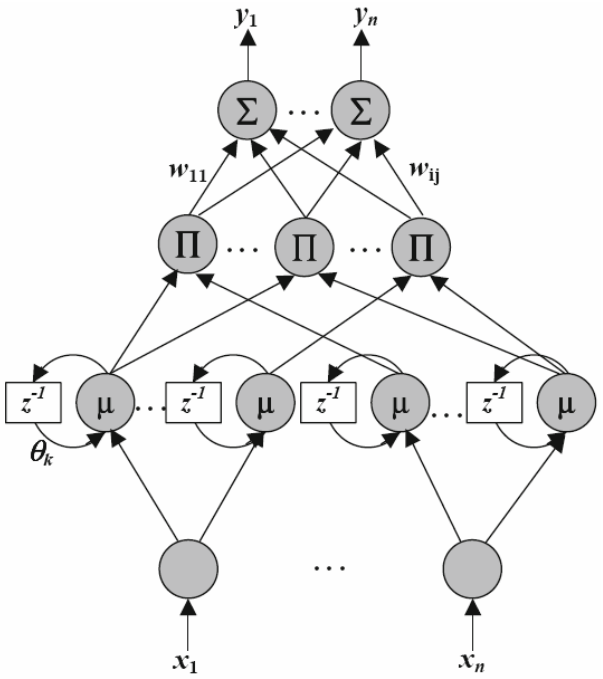


Fig. 7 Fuzzy neural network with simple feedback units as proposed by Lee and Teng

Approaches to learn hierarchical recurrent fuzzy system were presented in 2001 by Surmann and Maniadakis [65], who used a genetic algorithm, and Nürnberger [51], who proposed a template based approach to learn a structured rule base and a gradient descent based method motivated by the real time recurrent learning algorithm [76] to optimize the parameters of the learned rule base. The interpretability of the fuzzy sets of this model is ensured by the use of coupled weights in the consequents (fuzzy sets, which are assigned to the same linguistic terms share their parameters) and in the antecedents. Furthermore, constraints can be defined, which have to be observed by the learning method, e.g. that the fuzzy sets have to cover the considered state space. An example of the network structure is given in Fig. 8. However, the template based learning approach still had insufficiencies due to the use of a heuristic that created inner fuzzy sets. Therefore, in [52] a slightly modified approach was proposed, that learned the rule base using a genetic algorithm.

Furthermore, recurrent models that tackle specific problems of the learning process, properties of recurrent fuzzy systems or specific applications have been proposed in, e.g., [36, 72, 33].

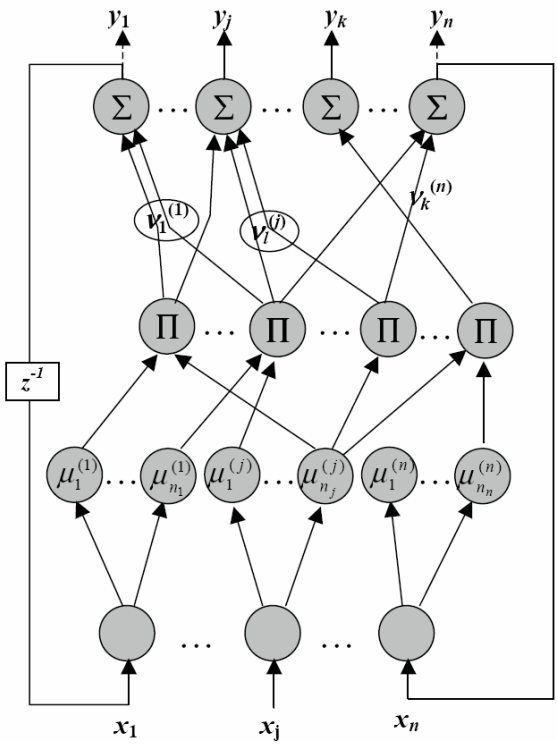


Fig. 8 Possible structure of the recurrent neuro-fuzzy system proposed by Nürnberger in 2001 [51] (using one time-delayed and one hierarchical feed-back). The first row of neurons defines the input variables, the second row the membership functions of the antecedents, the third row the fuzzy rules, and the fourth row the output variables. The membership functions of the consequents that are shared by rules are represented by coupled links from the rule to the output layer.

4 Outlook

Starting with neural network oriented architectures like ARIC and NNDFR neuro-fuzzy system quickly developed into network representations of fuzzy systems like we can see in ANFIS and NEFCON. In the second half of the 1990s we saw a lot of specific architectures for approximation, classification and control where neuro-fuzzy systems have covered a broad area of problems. Meanwhile, they found their way in quite diverse application areas where they are currently regularly applied, see e.g. recent works in geochemistry [81], geology [55], manufacturing [67], time series analysis [2] and signal processing [61].

However, there are still a lot of open research research questions in the area of adaptive control, where the combination of reinforcement learning methods with neuro-fuzzy architectures has made a lot of progress more recently (see, e.g., [16, 34, 71]). The same holds for applications in classification that became more

and more important with the growing interest in data mining (see, e.g., [1, 79]). In this area questions of how to completely automate the learning process and how to guarantee a certain level of interpretability remain to be important issues.

5 Remarks

We like to apologize to all researchers we did not mention in this — for this broad topic — short article. This would have been impossible. We tried only to mark major developments in this area and may have missed some that would be considered by others as major contributions.

References

[1] Asaithambi, M., Manoharan, S.C., Subramanian, S.: Classification of Respiratory Abnormalities Using Adaptive Neuro Fuzzy Inference System. In: Pan, J.-S., Chen, S.-M., Nguyen, N.T. (eds.) ACIIDS 2012, Part III. LNCS, vol. 7198, pp. 65–73. Springer, Heidelberg (2012)

[2] Atsalakis, G., Valavanis, K.: Forecasting stock market short-term trends using a neuro-fuzzy based methodology. Expert Systems with Applications 36(7), 10,696–10,707 (2009)

[3] Barto, A.G., Sutton, R.S., Anderson, C.W.: Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, and Cybernetics 13, 834–846 (1983)

[4] Berenji, H.R., Khedkar, P.: Fuzzy rules for guiding reinforcement learning. In: Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 1992), Mallorca, pp. 511–514 (1992a)

[5] Berenji, H.R., Khedkar, P.: Learning and tuning fuzzy logic controllers through reinforcements. IEEE Transactions on Neural Networks 3, 724–740 (1992)

[6] Berenji, H.R., Lea, R.N., Jani, Y., Khedkar, P., Malkani, A., Hoblit, J.: Space shuttle attitude control by reinforcement learning and fuzzy logic. In: Proc. IEEE Int. Conf. on Neural Networks 1993, San Francisco, pp. 1396–1401 (1993)

[7] Berthold, M.: Mixed fuzzy rule formation. Int. J. Approximate Reasoning 32, 67–84 (2003)

[8] Berthold, M., Huber, K.P.: Tolerating missing values in a fuzzy environment. In: Mares, M., Mesiar, R., Novak, V., Ramik, J., Stupnanova, A. (eds.) Proc. of 7th Intl. Fuzzy Systems Association World Congress (IFSA 1997), vol. I, pp. 359–362. Academia, Prague (1997)

[9] Berthold, M., Huber, K.P.: Constructing fuzzy graphs from examples. Int J Intelligent Data Analysis 3(1), 37–57 (1999)

[10] Berthold, M.R., Huber, K.P.: Neural network based construction of fuzzy graphs. In: Proceedings of the Fourth Annual Conference on Fuzzy Theory and Technology, North Carolina, pp. 170–173 (1995) (invited paper)

[11] Casillas, J., Cordon, O., Herrera, F., Magdalena, L. (eds.): Trade-off between Accuracy and Interpretability in Fuzzy Rule-Based Modelling. STUDFUZZ. Physica-Verlag, Heidelberg (2002)

[12] Dabija, V., Tschichold Gürman, N.: A framework for combining symbolic and connectionist learning with equivalent concept descriptions. In: Proc. 1993 Int. Joint Conf. on Neural Networks (IJCNN 1993), Nagoya (1993)
[13] Dias, P., Figueiredo, K., Vellasco, M., Pacheco, M.A., Barbosa, C.H.: Reinforcement learning hierarchical neuro-fuzzy model for autonomous robots. In: Proc. of 2nd Int. Conf. on Autonomous Robots and Agents (ICARA 2004), Palmerston, New Zealand, pp. 107–112 (2004)
[14] Eklund, P., Klawonn, F.: Neural fuzzy logic programming. IEEE Trans. Neural Networks 3, 815–818 (1992)
[15] Figueiredo, K., Vellasco, M., Pacheco, M., Souza, F.: Reinforcement learning hierarchical neuro-fuzzy politree model for control of autonomous agents. In: Proc. of 4th Int. Conf. on Hybrid Intelligent Systems (HIS 2004), Kitakyushu, Japan, pp. 107–112 (2004)
[16] Figueiredo, K., Campos, L.C.D., Vellasco, M.B.R., Pacheco, M.A.C.: Reinforcement learning-hierarchical neuro-fuzzy politree model for autonomous agents–evaluation in a multi-obstacle environment. In: Proc. of 5th Intl. Conf. on Hybrid Intelligent Systems (HIS 2005), pp. 551–554 (2005)
[17] Gorrini, V., Bersini, H.: Recurrent fuzzy systems. In: Proc. of the 3rd Conference on Fuzzy Systems (FUZZ-IEEE 1994). IEEE, Orlando (1994)
[18] Gupta, M., Rao, D.: On the principles of fuzzy neural networks. Fuzzy Sets and Systems 61, 1–18 (1994)
[19] Halgamuge, S.K.: Advanced methods for fusion of fuzzy systems and neural networks in intelligent data processing, PhD thesis, Technische Hochschule Darmstadt (1995)
[20] Halgamuge, S.K., Glesner, M.: A fuzzy-neural approach for pattern classification with the generation of rules based on supervised learning. In: Proc. Neuro-Nimes 1992, Nanterre, pp. 167–173 (1992)
[21] Halgamuge, S.K., Glesner, M.: Neural networks in designing fuzzy systems for real world applications. Fuzzy Sets and Systems 65, 1–12 (1994)
[22] Hammell II, R.J., Sudkamp, T.: A two level architecture for fuzzy learning. Journal of Intelligent and Fuzzy Systems 3(4), 273–286 (1995)
[23] Jang, J.S.R.: Fuzzy modeling using generalized neural networks and kalman filter algorithm. In: Proc. Ninth National Conf. on Artificial Intelligence (AAAI 1991), pp. 762–767 (1991)
[24] Jang, J.S.R.: Self-learning fuzzy controller based on temporal back-propagation. IEEE Transactions on Neural Networks 3, 714–723 (1992)
[25] Jang, J.S.R.: Anfis: Adaptive-network-based fuzzy inference systems. IEEE Transactions on Systems, Man, and Cybernetics 23, 665–685 (1993)
[26] Jang, J.S.R., Mizutani, E.: Levenberg-marquardt method for anfis learning. In: Proc. Biennial Conference of the North American Fuzzy Information Processing Society, NAFIPS 1996, pp. 87–91. IEEE, Berkeley (1996)
[27] Jang, J.S.R., Sun, C.T.: Functional equivalence between radial basis function networks and fuzzy inference systems. IEEE Trans. Neural Networks 4, 156–163 (1993)
[28] Keller, J.M.: Experiments on neural network architectures for fuzzy logic. In: Lea, R.N., Villareal, J. (eds.) Proc. Second Joint Technology Workshop on Neural Networks and Fuzzy Logic, NASA, Lyndon B, pp. 201–216. Johnson Space Center, Houston (1991)
[29] Keller, J.M., Hunt, D.J.: Incorporating fuzzy membership function into the perceptron algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence 7, 693–699 (1985)

[30] Keller, J.M., Tahani, H.: Backpropagation neural networks for fuzzy logic. Information Sciences 62, 205–221 (1992)
[31] Keller, J.M., Tahani, H.: Implementation of conjunctive and disjunctive fuzzy logic rules with neural networks. Int. J. Approximate Reasoning 6, 221–240 (1992)
[32] Keller, J.M., Yager, R.R., Tahani, H.: Neural network implementation of fuzzy logic. Fuzzy Sets and Systems 45, 1–12 (1992)
[33] Kempf, R., Adamy, J.: Equilibria of recurrent fuzzy systems. Fuzzy Sets and Systems 140(2), 231–257 (2003)
[34] Kuremoto, T., Yamano, Y., Feng, L.-B., Kobayashi, K., Obayashi, M.: A Neuro-fuzzy Network with Reinforcement Learning Algorithms for Swarm Learning. In: Zhang, Y. (ed.) Future Wireless Networks and Information Systems. LNEE, vol. 144, pp. 101–108. Springer, Heidelberg (2012)
[35] Lee, C.H., Teng, C.C.: Indentification and control of dynamic systems using recurrent fuzzy neural networks. IEEE Transactions on Fuzzy Systems 8(4), 349–366 (2000)
[36] Lin, C.M., Hsu, C.F.: Identification of dynamic systems using recurrent fuzzy neural network. In: Proc. of Joint 9th IFSA World Congress and 20th NAFIPS International Conference, pp. 2671–2675 (2001)
[37] Medsker, L.R., Jain, L.C.: Recurrent Neural Networks: Design and Applications. CRC Press (1999)
[38] Mitra, S., Hayashi, Y.: Neuro-fuzzy rule generation: survey in soft computing framework. IEEE Trans. Neural Netw. Learning Syst. 11(3), 748–768 (2000)
[39] Miyoshi, T., Tano, S., Kato, Y., Arnould, T.: Operator tuning in fuzzy production rules using neural networks. In: Proc. IEEE Int. Conf. on Fuzzy Systems 1993, San Francisco, pp. 641–646 (1993)
[40] Nauck, D.: Adaptive rule weights in neuro-fuzzy systems. Neural Computing & Applications 9(1), 60–70 (2000)
[41] Nauck, D.: Fuzzy data analysis with NEFCLASS. Int. J. Approximate Reasoning 32, 103–130 (2003)
[42] Nauck, D.: Measuring interpretability in rule-based classification systems. In: Proc. IEEE Int. Conf. on Fuzzy Systems 2003, pp. 196–201. IEEE, St. Louis (2003)
[43] Nauck, D.: Neuro-fuzzy learning with symbolic and numeric data. Soft Computing 8(6), 383–396 (2004)
[44] Nauck, D., Kruse, R.: Nefcon-i: An x-window based simulator for neural fuzzy controllers. In: Proc. IEEE Int. Conf. Neural Networks 1994 at IEEE WCCI 1994, Orlando, FL, pp. 1638–1643 (1994)
[45] Nauck, D., Kruse, R.: Nefclass–a neuro-fuzzy approach for the classification of data. In: George, K.M., Carrol, J.H., Deaton, E., Oppenheim, D., Hightower, J. (eds.) Proc. 1995 ACM Symposium on Applied Computing, Applied Computing 1995, Nashville, February 26-28, pp. 461–465. ACM Press, New York (1995)
[46] Nauck, D., Kruse, R.: A neuro-fuzzy method to learn fuzzy classification rules from data. Fuzzy Sets and Systems 89, 277–288 (1997)
[47] Nauck, D., Kruse, R.: How the learning of rule weights affects the interpretability of fuzzy systems. In: Proc. 7th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 1998), Anchorage, pp. 1235–1240 (1998)
[48] Nauck, D., Kruse, R.: Neuro-fuzzy systems for function approximation. Fuzzy Sets and Systems 101, 261–271 (1999)
[49] Nauck, D., Nürnberger, A.: The evolution of neuro-fuzzy systems. In: Proc. 24th International Conf. of the North American Fuzzy Information Processing Society (NAFIPS 2005), pp. 98–103. IEEE (2005)

[50] Nauck, D., Klawonn, F., Kruse, R.: Foundations of Neuro-Fuzzy Systems. Wiley, Chichester (1997)
[51] Nürnberger, A.: A hierarchical recurrent neuro-fuzzy system. In: Proc. of Joint 9th IFSA World Congress and 20th NAFIPS International Conference, pp. 1407–1412. IEEE (2001)
[52] Nürnberger, A.: Approximation of dynamic systems using recurrent neuro-fuzzy techniques. Soft Computing 8(6), 428–442 (2004)
[53] Nürnberger, A., Nauck, D., Kruse, R.: Neuro-fuzzy control based on the nefcon-model. Soft Computing 2(4), 182–186 (1999)
[54] Pal, S.K., Mitra, S.: Multi-layer perceptron, fuzzy sets and classification. IEEE Trans. Neural Networks 3, 683–697 (1992)
[55] Park, I., Choi, J., Lee, M.J., Lee, S.: Application of an adaptive neuro-fuzzy inference system to ground subsidence hazard mapping. Computers and Geosciences (2012) (online first)
[56] Pedrycz, W.: Neurocomputations in relational systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 13, 289–297 (1991)
[57] Pedrycz, W.: A referential scheme of fuzzy decision making and its neural network structure. IEEE Transactions on Systems, Man, and Cybernetics 21, 1593–1604 (1991)
[58] Pedrycz, W., Kandel, A., Zhang, Y.Q.: Neurofuzzy systems. In: Nguyen, H.T., Sugeno, M. (eds.) Fuzzy Systems Modeling and Control. The Handbooks on Fuzzy Sets ch. 9, pp. 311–380. Kluwer Academic Publishers, Norwell (1998)
[59] Pineda, F.J.: Recurrent backpropagation networks. In: Chauvin, Y., Rumelhart, D.E. (eds.) Backpropagation: Theory, Architectures and Applications, pp. 99–135. Lawrence Erlbaum Associates, Hillsdale (1995)
[60] Riedmiller, M.: Selbständig lernende neuronale Steuerungen. VDI-Verlag GmbH, Düsseldorf (1997)
[61] Sarma, K.K., Mitra, A.: Recurrent fuzzy-neural mimo channel modeling. Journal of Intelligent Systems (2012)
[62] Simpson, P.K.: Fuzzy min-max neural networks–part 1: Classification. IEEE Transactions on Neural Networks 3, 776–786 (1992a)
[63] Simpson, P.K.: Fuzzy min-max neural networks–part 2: Clustering. IEEE Transactions on Fuzzy Systems 1, 32–45 (1992)
[64] Sudkamp, T., Hammell II, R.J.: Scalability in fuzzy rule-based learning. Information Sciences 199, 135–147 (1998)
[65] Surmann, H., Maniadakis, M.: Learning feed-forward and recurrent fuzzy systems: A genetic approach. Journal of Systems Architecture 47(7), 649–662 (2001)
[66] Takagi, H., Hayashi, I.: NN-driven fuzzy reasoning. Int. J. Approximate Reasoning 5, 191–212 (1991)
[67] Taylan, O., Darrab, I.A.: Fuzzy control charts for process quality improvement and product assessment in tip shear carpet industry. Journal of Manufacturing Technology Management 23(3), 402–420 (2012)
[68] Theocharis, J.B., Vachtsevanos, G.: Recursive learning algorithms for training fuzzy recurrent models. International Journal of Intelligence Systems 11(12), 1059–1098 (1996)
[69] Tschichold-Gürman, N.: Generation and improvement of fuzzy classifiers with incremental learning using fuzzy rulenet. In: George, K.M., Carrol, J.H., Deaton, E., Oppenheim, D., Hightower, J. (eds.) Proc. 1995 ACM Symposium on Applied Computing, Applied Computing, Nashville, February 26-28, pp. 466–470. ACM Press, New York (1995)

[70] Tschichold-Gürman, N.: Rulenet–a new knowledge-based artificial neural network model with application examples in robotics. PhD thesis, ETH Zürich, Zürich (1996)
[71] Vatankhah, R., Etemadi, S., Alasty, A., Vossoughi, G.: Adaptive critic-based neuro-fuzzy controller in multi-agents: Distributed behavioral control and path tracking. Neurocomput. 88, 24–35 (2012)
[72] Wang, J.S., Lee, C.S.G.: Self-adaptive recurrent neuro-fuzzy control of an autonomous underwater vehicle. IEEE Transactions on Robotics and Automation 19(2), 283–295 (2003)
[73] Wang, L.X., Mendel, J.M.: Generating fuzzy rules by learning from examples. IEEE Trans Syst., Man, Cybern. 22(6), 1414–1427 (1992)
[74] Wermter, S.: Neural fuzzy preference integration using neural preference moore machines. International Journal of Neural Systems 10(4), 287–310 (2000)
[75] White, D.A., Sofge, D.A. (eds.): Handbook of Intelligent Control. Neural, Fuzzy, and Adaptive Approaches. Van Nostrand Reinhold, New York (1992)
[76] Williams, R.J., Zipser, D.: Experimental analysis of the real time recurrent learning algorithm. Connection Science 1, 87–111 (1989)
[77] Yager, R.R., Filev, D.P.: Adaptive defuzzification for fuzzy logic controllers. BUSEFAL 49, 50–57 (1992)
[78] Yager, R.R., Filev, D.P.: Adaptive defuzzification for fuzzy system modelling. In: Proc. Workshop of the North American Fuzzy Information Processing Society (NAFIPS 1992), pp. 135–142. Puerto Vallarta (1992)
[79] Zhang, C., Qiu, F.: Hyperspectral image classification using an unsupervised neuro-fuzzy system. Journal of Applied Remote Sensing 6(1), 063, 515 (2012)
[80] Zhang, J., Morris, A.J.: Recurrent neuro-fuzzy networks for nonlinear process modelling. IEEE Transactions on Neural Networks 10(2), 313–326 (1999)
[81] Ziaii, M., Ardejani, F.D., Ziaei, M., Soleymani, A.A.: Neuro-fuzzy modeling based genetic algorithms for identification of geochemical anomalies in mining geochemistry. Applied Geochemistry 27(3), 663–676 (2012)

Safe and Interpretable Machine Learning: A Methodological Review

Clemens Otte

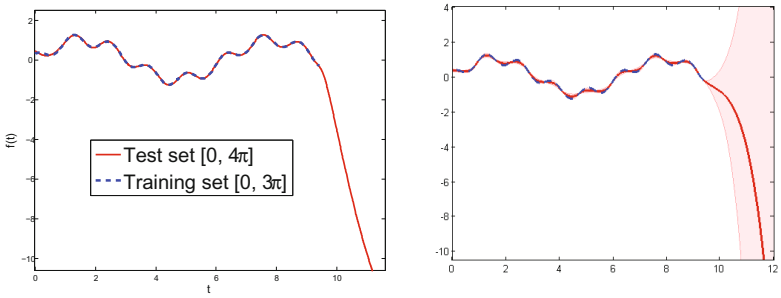
Abstract. When learning models from data, the interpretability of the resulting model is often mandatory. For example, safety-related applications for automation and control require that the correctness of the model must be ensured not only for the available data but for all possible input combinations. Thus, understanding what the model has learned and in particular how it will extrapolate to unseen data is a crucial concern. The paper discusses suitable learning methods for classification and regression. For classification problems, we review an approach based on an ensemble of nonlinear low-dimensional submodels, where each submodel is simple enough to be completely verified by domain experts. For regression problems, we review related approaches that try to achieve interpretability by using low-dimensional submodels (for instance, MARS and tree-growing methods). We compare them with symbolic regression, which is a different approach based on genetic algorithms. Finally, a novel approach is proposed for combining a symbolic regression model, which is shown to be easily interpretable, with a Gaussian Process. The combined model has an improved accuracy and provides error bounds in the sense that the deviation from the verified symbolic model is always kept below a defined limit.

1 Introduction

There is an increasing trend for using data-driven models (i.e. models learned from data) in monitoring and control applications, for instance in industry, healthcare or automotive electronics [9, 18]. The main reason usually is that analytical models derived from first principles are either unknown or suffer from insufficient accuracy. However, deploying data-driven models in applications where incorrect model outputs may have fatal consequences requires ensuring that the model is correct for all possible inputs. In practice, training data are almost always limited and may not

Clemens Otte
Siemens AG, Corporate Research and Technologies, 81739 Munich, Germany
e-mail: clemens.otte@siemens.com

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 111–122.
springerlink.com



(a) Single neural network (b) Ensemble of 50 networks

Fig. 1 Example of inappropriate extrapolation behavior. Left side: Neural network is tested outside the interval used for training. Right side: Uncertainty of the model is estimated by averaging the output of 50 neural networks. The uncertainty is large beyond the training data (shading shows mean ± 1 standard deviation)

represent all relevant operating conditions. Thus, it is crucial to understand what the model has learned and in particular how it will extrapolate to unseen data.

A simple example is shown in Fig. 1(a). A feed-forward network [2] with a single input, one hidden layer of nine neurons with sigmoidal activation function and a single output with linear activation has been trained on 1000 noise-free training samples drawn from a sum of two sines on the interval $[0, 3\pi]$. After training, it is tested on the larger interval $[0, 4\pi]$. While the test accuracy on the interval seen during training is very good, the extrapolation behavior is rather inappropriate. It is obvious that a linear combination of sigmoids cannot reproduce the periodicity of the true function. The key point here is that the model is tested in a range where the desired model behavior is not specified by training data. In other words, the model has learned an approximation to the unknown true function on the training data and beyond that data the approximation is poor.

A possible way for estimating the model uncertainty is to consider the average and standard deviation of the output of several models as shown in Fig. 1(b). In this example each of 50 neural networks was initialized with different random weights. They converged to slightly different solutions which are similar on the training data but have diverse extrapolation behavior. The resulting standard deviation is large in the region not specified by the training data, which might be used to alarm the user about the uncertainty. A further discussion of model averaging can be found in [5, 7].

While model averaging is a common approach for improving the prediction accuracy, there is no guarantee that the uncertainty estimation is correct. This is due to the fact that the ensemble members usually are not truly independent because, for example, they are trained on the same data or are based on the same type of model.

Other approaches for avoiding inappropriate extrapolation behavior include the following ones.

- Use of an additional model estimating the density in the input space and deactivating the prediction model in low-density regions. However, density estimation in higher-dimensional spaces is a challenge itself [7, 17].
- Constraining the model using a-priori knowledge. For example, if some input-output relations are known to be monotonic then monotonicity constraints can be introduced on the weights of a neural network [8].
- Limiting the allowed output range of the model, e.g. to the range given by the training data. This is of course just an ad hoc method.

None of these approaches can in general guarantee that unspecified behavior of the model does not occur. It is therefore preferable to use models possessing a level of interpretability that suits the safety requirements of the respective application. Usually, one has to weight interpretability against accuracy, so there is no magic formula. Instead, among the variety of methods the best one has to be chosen specific to the application. Some examples are given in this paper.

Structure of this Paper

The next section describes an approach for classification problems with an application to airbag control in automotive safety. Some basic ideas are then transferred to the learning of regression functions in Sect. 3. Using a benchmark data set (SARCOS inverse dynamics problem) several regression approaches are discussed. It is shown that a combination of symbolic regression with a Gaussian Process (GP) provides a good trade-off between interpretability and accuracy. In that combination, symbolic regression provides an analytical model and the GP improves the overall accuracy by learning the residuals of the analytical model. Section 4 concludes.

2 Safe and Interpretable Classification

In the following we review an approach based on an ensemble of nonlinear low-dimensional submodels where each submodel is simple enough to be completely verified by domain experts. The approach was firstly developed for binary classification problems [12] and then extended to multiple classes [13].

2.1 Ensembles of Low-Dimensional Submodels

The algorithm for learning an ensemble of low-dimensional submodels for binary classification problems is described in plain words below. For a formal description and extension to more than two classes we refer to [12, 13].

1. Given: Data set D with feature vectors in $\mathbb{R}^p$ of two classes (positive and negative class). Start with an empty set of selected models.
2. Consider all two-dimensional subspaces and learn a separation of both classes in each subspace, e.g. by a support-vector classifier, neural network or any other method. There is one important constraint for learning: The model must correctly

classify all negative samples in D, that is, false positives (i.e. false alarms) must not occur.
3. Select the best 2D-model (possibly involving expert knowledge in the selection process) and add the model to the set of selected models.
4. Remove all correctly classified positive samples from D.
5. Start again in step 2 to separate remaining positive samples until no positive instances remain or the maximum size of the set of selected models has been reached.

The reason for the no-false-alarm requirement in the second step is that it allows combining all selected models simply by logical-or. In other words a test sample will be assigned to the positive class if at least one of the models assigns it to that class. This greatly improves the interpretability of the ensemble as shown in the following example.

2.2 Example: Airbag Control

In this safety-relevant application the objective is to learn the deployment decision of an airbag system from crash test data. In each crash test the data from several sensors in the car (mostly acceleration sensors) were recorded and various features were extracted, resulting in a multivariate time-series sampled every 1 ms. In our experiment the data set includes 52 features. Each crash test either has a "nofire" or "fire" label. In case of a "fire" label, the crash has a desired point of time when the airbag has to be triggered, e.g. 24 ms after the first contact with the barrier. A "nofire" crash must not trigger the airbag at any time.

Since the time series of each crash comprises many samples that all share the same label, this is a typical example of a class of problems known as multiple-instance-learning [1]. For these problems it is sufficient to classify at least one sample of a "fire" crash correctly (within a certain time interval around the desired fire time). In contrast, no sample of a "nofire" crash may ever be misclassified as "fire".

Figure 2 shows three models learned by the algorithm. Note that the training data only cover some part of the input space. However, by using two-dimensional projections and visualizing the class boundary in the whole 2D-space, the extrapolation behavior can be verified. The features of each model are selected from the set of $\binom{52}{2} = 1326$ combinations of the original, physically interpretable features.

3 Safe and Interpretable Regression

As explained above in the context of Fig. 1 it is difficult and often practically impossible to understand a neural network model on such a detailed level that it would be possible to make statements about its extrapolation behavior. The same holds for many other successful approaches like Support Vector Machines [16] and Gaussian Processes [14].

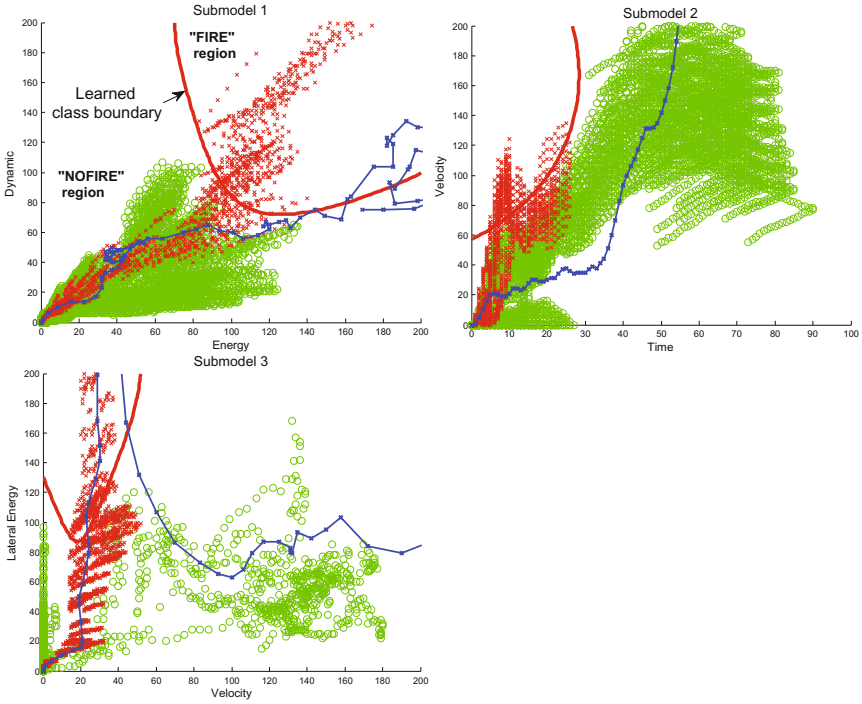


Fig. 2 Three airbag control submodels, each separating a certain "fire" region from a "nofire" region in a 2D-projection of the input space. Training samples from "nofire" crashes are marked as (green) circles. Samples from "fire" crashes are marked as (red) crosses. The thick line indicates the learned class boundary in each model. By explicitly visualizing the boundary in the complete input space of each model the extrapolation behavior can be verified. Note that large regions of the input space are not filled by data. Thus, manually checking the extrapolation behavior is mandatory. For illustration, the trajectory of a particular "fire" crash is shown in the respective 2D-projections. It starts in the origin and enters the "fire" region in the third model (second row, left) and some time later also in the first model (first row, left). A "fire" decision by one model is sufficient.

Traditionally, approaches like the **C**lassification **A**nd **R**egression **T**rees (CART) [4] or rule-based methods [11] are considered as being interpretable. There is of course a trade-off with accuracy. For example, a CART model only remains interpretable as long as the number of tree nodes is rather small, which means that the model is quite coarse. In contrast, Random Forests [3] provide the other extreme with a good reputation in terms of accuracy but without any kind of interpretability.

A possible compromise is to partition the input space similar to CART or rule learners but to use slightly more complex submodels in the different regions of the input space. MARS (multivariate adaptive regression splines) [6] and an approach called GUIDE (generalized unbiased interaction detection and estimation) [10] follow this strategy; both are discussed in an experiment below.

Symbolic regression [15] comes from a different direction and seeks for a symbolic representation (i.e. an equation) that best matches the given data. Details are given later on.

In statistics, additive models (see e.g. chapter 9 in [7]) are often applied when the model shall be interpretable. The idea is to learn a multivariate function having p inputs as a sum of p univariate functions, that is, $y = \alpha + \sum_{i=1}^{p} f_i(x_i) + \varepsilon$. The advantage is that the univariate functions may be easier to interpret. The drawback, however, is that interactions between variables cannot be modeled.

In the experiment below we propose a different strategy which is similar to an additive model in the sense that two modular functions are added in the final model. The first function is used as an analytical model learned by symbolic regression; it is easily interpretable but has only moderate accuracy. Thus, a second function is learned on the residuals of the first function in order to improve the accuracy. The model of the second function is not interpretable; but by limiting its output to a defined range a worst-case guarantee can be given in the sense that the maximal deviation from the analytical model is always below a certain limit.

The objective of the following experiment is to compare different regression approaches in terms of their accuracy and interpretability. The results are discussed in Sect. 3.2.

3.1 Experiment: SARCOS Benchmark

In this example we consider learning the inverse dynamics of a seven degrees-of-freedom SARCOS robot arm[1]. The task is to map from a 21-dimensional input space (7 joint positions, 7 joint velocities, 7 joint accelerations) to the corresponding 7 joint torques. Following previous studies on this benchmark (see references in [14]) we only consider the mapping to the first of the seven torques, that is, we learn a function $f : \mathbb{R}^{21} \to \mathbb{R}$. There are 44,484 training examples and 4,449 test examples. All 21 inputs x_i have been standardized to have zero mean and standard deviation 1. The output y has zero mean. Results are given as *standardized mean squared error* (SMSE), which is the mean squared error on the test set divided by the variance of the target values in the test set. The normalization by the variance makes the error measure independent on the overall scale of the target.

While in a real safety-related application the *worst-case error* should be considered additionally to the mean error, here we solely use the SMSE to facilitate the comparison with previous studies.

All methods described in the following were applied to this benchmark. The resulting SMSE accuracy is summarized in Table 1.

Rigid-body-dynamics (RBD): This is a physics-based model derived from rigid-body-dynamics. The RBD result is taken from [14], p. 24. As shown in Table 1 the model accuracy is rather poor, which may be explained by the fact that the robot

[1] Named after the robotics company SARCOS. The benchmark data are publicly available from `http://www.gaussianprocess.org/gpml/data/`

is actuated hydraulically and is rather lightweight, so some rigid-body assumptions seem to be violated.

Linear Regression (LR): Linear model with 21 inputs and no intercept.

GUIDE [10]: Similar to the classical CART this is a recursive partitioning algorithm creating a regression tree. In CART a leaf of the regression tree contains just the mean output value of all training samples assigned to that leaf. In contrast, in GUIDE the leaves may contain linear models either with all inputs or with a subset of inputs. With the complexity of the submodels in the leaves being larger in GUIDE, the overall number of nodes can often be kept smaller in comparison to CART, which may improve the interpretability of the final model.

Note that switching between nodes may lead to discontinuities in the output, possibly causing problems in control applications. This holds for other approaches as well that use a hard partitioning of the input space, e.g. CART, MARS.

Three experiments were conducted. They differ in the type of linear models used in the terminal nodes (leaves). In the second experiment the minimum node size was raised; the parameter defines the minimum number of training samples that a node must have. Increasing this number reduces the number of nodes and helps in improving the interpretability. The lowest error is achieved in the third experiment, which is the value considered in Table 1. However, the model is not easy to interpret: it consists of 35 linear models each having 21 variables.

Experiment 1: Stepwise linear models with up to 5 variables. Using defaults (minimum node size = 889). Number of terminal nodes of final tree: 27, SMSE = 0.0411

Experiment 2: Stepwise linear models without limiting the number of variables (typically 11-17 variables entered the models in the leaves). Minimum node size set to 2000. Number of terminal nodes of final tree: 16, SMSE = 0.0403

Experiment 3: Multiple linear models, all with 21 variables, using defaults (minimum node size = 889). Number of terminal nodes of final tree: 35, SMSE = 0.0327

MARS: Multivariate adaptive regression splines [6] are an approach where a model is build as a linear combination of basis functions. The simplest basis function is a piecewise linear function (linear spline) $h(x_i)$ of one input variable; instead of linear splines cubic splines may also be used. Interactions between two inputs x_i and x_j are modeled by the product $h(x_i)g(x_j)$ of two univariate spline functions. The resulting product is considered as a basis function again. Higher-order interactions can be handled analogously by building the respective products. Usually the maximum interaction level is limited to aid in the interpretation of the final model. A key property of the basis functions is that they are zero over some part of their range, making it possible for them to operate locally.

We used the ARESLab[2] toolbox ver. 1.5.1 in Matlab. The maximum number of basis functions allowed to be included in the model was set to 21 (including the

[2] ARESLab obtainable from http://www.cs.rtu.lv/jekabsons/

intercept term) and the maximum interaction level was limited to 2 (only allowing pairwise products). The resulting model trained on the complete training set is shown below.

```
y = 21.51 +22.19*BF1 -25.44*BF2 +4.49*BF3 +5.88*BF4 +12.61*BF5 -8.45*BF6 +3.03*BF7 -14.14*BF8
+2.31*BF9 +1.63*BF10 +7.37*BF11 +2.69*BF12 -4.02*BF13 -3.20*BF14 -1.88*BF15 -2.37*BF16 -2.11*BF17
-4.43*BF18 +1.98*BF19 +1.60*BF20
```

where all basis functions are either univariate cubic splines or products of two univariate cubic splines. The model is rather difficult to interpret.

Gaussian Process (GP): A GP is a linear smoother using a weighted average of the stored training outputs $\mathbf{y}$ to predict the output for a test input. It can be seen as a linear combination of n kernel functions, each one centered on one of n training points [14],

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}), \quad \alpha = (K + \sigma_n^2 I)^{-1}\mathbf{y} \tag{1}$$

where $\mathbf{x}$ is a test input, $\mathbf{x}_i$ are the training inputs and α is a weight vector with kernel matrix K and a hyperparameter σ_n.

We used the squared exponential kernel where each input dimension is scaled by an individual factor, allowing the down-weighting of irrelevant inputs. The kernel function is given as

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T D^{-2} (\mathbf{x} - \mathbf{x}')\right)$$

with diagonal matrix $D = \mathrm{diag}(\ell_1, \ldots, \ell_{21})$ containing the scaling factors of the input dimensions. In total 23 hyperparameters $(\ell_1, \ldots, \ell_{21}, \sigma_f, \sigma_n)$ were optimized during training.

Note that the GP cannot be trained on the complete training set as the kernel matrix would be too large to compute the inverse in Eq. (1). Thus, we randomly draw a subset of 4000 samples from the original training set and trained the GP on the reduced set. The GP was then applied to the test set and the total procedure was repeated ten times. Table 1 shows the average and standard deviation of the ten runs. The SMSE is in good accordance to the result reported in [14], p. 182. It is possible to slightly reduce the error further by replacing the random subsampling with more sophisticated methods. This has not been investigated in this paper, for details we refer to [14].

In terms of the interpretability of the model the GP has to be considered as a "black box" approach as the model of Eq. (1) is a linear combination of 4000 kernel functions.

Eureqa: Eureqa is a tool for symbolic regression[3], where the training data are used to search for an explicit symbolic relationship of the form $y = f(\mathbf{x})$. Based on genetic programming the search space of equations consisting of pre-specified "building

[3] http://creativemachines.cornell.edu/eureqa

blocks" (e.g. arithmetic operators, sin, exp) is explored to find an equation best matching the data [15]. We used the mean-squared error to guide the search on the complete training set and took the following model having the smallest training set error.

$$\begin{aligned} &\text{y = x2 + 25.31*x15 + 11.08*x1 + 11.08*x18 + 1.732*x21 + x1*x15 + x4*x21} \\ &\text{- x11 - x2*x15 - x4*x18 - 1.732*x8*x9} \end{aligned} \tag{2}$$

Note that this representation is more compact and much more interpretable than the MARS model. Unlike the LR model the Eureqa model includes several terms with two interacting variables, so it is not surprising that the model achieves a better test set SMSE than the LR model as shown in Table 1.

A certain drawback of symbolic regression is that it is computationally intensive; we spent about 60 hours search time on a 16 CPU cores computer. After ≈ 35 hours there were only little improvements so the search could have been stopped earlier. Given the much better interpretability of Eq. (2) in comparison to the GUIDE and MARS models, the higher search time may be acceptable.

Eureqa + GP: Here we propose a new approach where a Gaussian Process (GP) is used to learn the residuals of the Eureqa model. The basic idea is to take the Eureqa model as an easily verifiable analytical model and to improve the accuracy by a Gaussian Process. Thus, the overall model has the form $y = f(\mathbf{x}) + r(\mathbf{x})$ where f is the Eureqa equation and r is the GP model. The model r was trained on $n = 4000$ randomly drawn residuals $y^{tar} - f(\mathbf{x})$ where y^{tar} are the target output values of the training set. The experiment was repeated 10 times to avoid a sampling bias. The number n controls the accuracy of the GP residual model as in the case of the pure GP. A much smaller number of training samples, e.g. $n = 2000$, would yield a coarser model with less accuracy.

Note that in terms of safety little is gained unless the amount of correction r is limited. Otherwise, the output of the overall model may be arbitrarily wrong because it is hardly possible to fully verify the GP model as explained in the GP section above. Therefore, the output r is limited, giving the overall model $y = f(\mathbf{x}) + r^*(\mathbf{x})$ with $r^*(\mathbf{x}) = \max(\min(r(\mathbf{x}), \gamma), -\gamma)$. The limit $\gamma \in \mathbb{R}$ provides a worst-case guarantee on the error of the overall model. When choosing the limit, there is a trade-off between possible accuracy gains and the worst-case guarantee.

In the SARCOS training set the output is in the range $[-108.5, 107.7]$. We therefore evaluated the following limits $\gamma \in \{5, 10, 15, \infty\}$. For example, $\gamma = 10$ means that the maximal possible deviation from the analytical model is ± 10, which is less than ten percent of the maximal output values. We also tested the unlimited case, which is denoted as $\pm\infty$.

As shown in Table 1 the tightest limit ± 5 already improves the accuracy considerably in comparison to the pure Eureqa model. The error is reduced further when the limits are relaxed and at ± 15 the same accuracy as of the pure GP is achieved. Results show average and standard deviation over 10 runs.

Table 1 Results on the inverse dynamics problem. The error is given as standardized-mean-squared error (SMSE) on the test set. The third column refers to the interpretability of the resulting model.

Method	SMSE	Interpretability
RBD	0.104	very high
LR	0.075	high
GUIDE	0.033	moderate
MARS	0.059	moderate
GP	0.020 ± 0.001	very low
Eureqa	0.062	very high
Eureqa + GP (± 5)	0.028 ± 0.000	high
Eureqa + GP (± 10)	0.021 ± 0.001	high
Eureqa + GP (± 15)	0.020 ± 0.001	(high)
Eureqa + GP ($\pm \infty$)	0.020 ± 0.001	very low

3.2 Discussion

The best accuracy in this benchmark is achieved by Gaussian Processes (GP). However, as seen in Eq. (1) a GP consists of in our case 4000 kernel functions centered on the respective training points, so the model is not interpretable. In principle, a GP is able to additionally provide confidence estimates; but these may be wrong, for example, if the model is slightly misspecified. Therefore, it is problematic to apply a pure GP in safety-related applications.

The best interpretability among the data-driven approaches is achieved by the symbolic regression model of Eureqa. A domain expert should be able to verify the learned equation. In terms of accuracy the Eureqa model is better than the linear regression model (LR) but worse than the more complex models of MARS or GUIDE. However, the latter two are difficult to interpret.

The physics-based RBD model shows the worst accuracy on the test data, possibly due to violations of rigid-body assumptions. This clearly shows the improvements that are possible by using data-driven models.

The proposed combination of the Eureqa model with a GP greatly improves the accuracy. Limiting the possible contribution of the GP to a certain range gives a worst-case guarantee in the sense that the maximal deviation from the verified analytical model is always below that limit. Thus, the proposed approach may be very interesting for safety-related applications where these kinds of guarantees are mandatory.

The combination scheme can also be translated to other methods. For example, instead of using a GP, the residuals could be learned by support vector regression [16], which may be beneficial due to the automatic selection of support vectors. On the other hand, the verified model could be a rule-based or tree-based model instead of the symbolic regression equation as long as the model is truly interpretable.

4 Conclusions

Understanding what a model has learned and in particular how it will extrapolate to unseen data becomes a crucial concern if the model correctness must be ensured not only for the available data but for all possible input combinations. The paper has reviewed an approach for learning a classifier for safety-related applications. The basic idea in that case is to solve the problem by splitting it into several low-dimensional subproblems and to visualize the decision boundary of the classifier in the whole low-dimensional input space of the respective submodels.

For regression problems similar concepts exist. MARS and the tree-growing approach GUIDE use submodels which are valid only in certain regions of the input space. They have been compared together with other regression methods on the SARCOS data benchmark. In that application both MARS and GUIDE provide only moderate interpretability due to a large number of submodels. An excellent interpretability is achieved by symbolic regression; however, this comes with the price of a reduced accuracy. A good trade-off is provided by combining the symbolic model with a Gaussian Process that learns the residuals of the symbolic model. The output of the Gaussian Process model, that is, the amount of correction applied to the symbolic model is limited to a certain range. The combined model has an improved accuracy and provides error bounds in the sense that the deviation from the verified symbolic model is always kept below a defined limit.

References

[1] Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: Advances in Neural Information Processing Systems, vol. 15, pp. 561–568. MIT Press, Cambridge (2003)

[2] Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)

[3] Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)

[4] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Statistics/Probability Series. Wadsworth Publishing Company, Belmont (1984)

[5] Dietterich, T.G.: Ensemble Methods in Machine Learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)

[6] Friedman, J.H.: Multivariate Adaptive Regression Splines. The Annals of Statistics 19(1), 1–67 (1991)

[7] Hastie, T., Tibshirani, R., Friedman, J., Franklin, J.: The elements of statistical learning: data mining, inference and prediction. Springer, New York (2009), http://www-stat.stanford.edu/~tibs/ElemStatLearn

[8] Lang, B.: Monotonic Multi-layer Perceptron Networks as Universal Approximators. In: Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S. (eds.) ICANN 2005. LNCS, vol. 3697, pp. 31–37. Springer, Heidelberg (2005), doi:10.1007/11550907

[9] Lisboa, P.J.G.: Industrial use of safety-related artificial neural networks. Contract research report 327/2001. Liverpool John Moores University (2001)

[10] Loh, W.: Regression by parts: Fitting visually interpretable models with guide. In: Chen, C., Härdle, W., Unwin, A. (eds.) Handbook of Computational Statistics, pp. 447–468 (2008)

[11] Mitra, S., Hayashi, Y.: Neuro-fuzzy rule generation: Survey in soft computing framework. IEEE Transact. Neural Networks, 748–768 (2000)
[12] Nusser, S., Otte, C., Hauptmann, W.: Interpretable ensembles of local models for safety-related applications. In: Proceedings of 16th European Symposium on Artificial Neural Networks (ESANN 2008), Brugge, Belgium, pp. 301–306 (2008)
[13] Nusser, S., Otte, C., Hauptmann, W., Kruse, R.: Learning verifiable ensembles for classification problems with high safety requirements. In: Wang, L.S.L., Hong, T.P. (eds.) Intelligent Soft Computation and Evolving Data Mining: Integrating Advanced Technology, pp. 405–431. IGI Global (2009)
[14] Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2006)
[15] Schmidt, M., Lipson, H.: Distilling Free-Form Natural Laws from Experimental Data. Science 324(5923), 81–85 (2009)
[16] Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2002)
[17] Silverman, B.: Density Estimation for Statistics and Data Analysis (Chapman & Hall/CRC Monographs on Statistics & Applied Probability). Chapman and Hall/CRC (1986)
[18] Taylor, B.J. (ed.): Methods and Procedures for the Verification and Validation of Artificial Neural Networks. Springer (2005)

Science Visions, Science Fiction and the Roots of Computational Intelligence

Rudolf Seising

Abstract. In later science fiction movies, computers run countries or govern the whole mankind but in science fiction stories of the 1950s this scenario does not exist. It seems that it originated from the early Computer Science and it was Lotfi A. Zadeh who published in 1950 the first science vision of a "Thinking Machine". He also predicted in 1950 that "Thinking machines" may be commonplace in anywhere from ten to twenty years hence and that they will play a major role in any armed conflict. Not many years later new SF stories told these kinds of stories of computers that govern the world by their decision — sometimes they annihilate the earth, sometimes they protect the planet. This paper gives a historical view on the idea of "machines that/who thinks" in science visions and in science fiction. Then, it shows this idea's historical path from the research program of Artificial Intelligence to that of Computational Intelligence.

1 Introduction

Is there a difference between "Machines that compute" and "Machines that think"? — We are tempted to say that the answer is obviously "Yes!" and perhaps many of us could start to list some distinctive features immediately. However, this is a result from 20th century science and technology and its reflections in science and science fiction literature. Both will be traced in this chapter.

There was a time when both "Computing Machines" and "Thinking Machines" have been names for the same! — The buzz word "Thinking Machine" appeared in popular journals and newspapers, in science fiction stories, but also in scientific articles: Shortly after the end of World War II the general public became informed on the war development of computer and communication technology and a big number of headlines said that these new created machines were "mechanical"

Rudolf Seising
European Centre for Soft Computing, Edificio de Investigación, 33600 Mieres (Asturias), Spain
e-mail: rudolf.seising@softcomputing.es

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 123–150.
springerlink.com

(*The Baltimore Sun*) or "electronical" (*New York Herald Tribune*) or "mathematical brains" (*Philadelphia Inquire*), but they were also named "wonder brains" (*Philadelphia Inquire*), "magic brains" (*New York World Telegram*) and "superbrains" (*Newark Star Ledger*). They notified that these apparatus had a weight of 30 tons (*The Evening Bulletin, Providence*), they were 1000 times faster than any previously built (*Chicago Sun*) and they could compute a 100-year problem in two hours (*New York Herald Tribune*) ([22], p. 120.).

The first book that continued this trend was *Giant Brains or Machines that Think*. It was published by the mathematician Edmund C. Berkeley (1923–1988) in 1949. In this book Berkeley gave a description of the functionality of the early computing machines — the book cover promoted that "an authority tells the story of 'mechanical brains' — how they 'think', what they do and what they can mean in your future."[5]

"Can machines think?" was also the question that interested the British mathematician Alan M. Turing (1912-1954) in his article "Computing Machinery and Intelligence" that appeared one year later [47]. However, Turing started as follows: "I propose to consider the question, 'Can machines think?"' But: Since "thinking" is difficult to define, he chose to "replace the question by another, which is closely related to it and is expressed in relatively unambiguous words." ([47], p. 433) Now, Turing considered the question "Are there imaginable digital computers which would do well in the imitation game?" ([47], p. 442) that — as he believed — was one that could actually be answered and to this end he proposed the "imitation game" that was later named the "Turing test". Thus, there was no statement in Turing's paper to decide whether a computer or a program could think like a human being or not.

Being unaware of Turing's article, but inspired by Norbert Wiener's (1894–1964) *Cybernetics* [48], Claude E. Shannon's (1916–2001) "Mathematical Theory of Communication" [44] and the computer era that started during the wartime with the *Electronic Numerical Integrator and Computer* (ENIAC) and the *Electronic Discrete Variable Computer* (EDVAC) (both designed by John P. Eckert (1919-1995) and John W. Mauchly (1907–1980)), Lotfi A. Zadeh wrote in the same year the article "Thinking Machines: A New Field in Electrical Engineering."

Zadeh, born 1921 in Baku, Azerbaijan, had studied and graduated with a Bachelor of Science in electrical engineering from the University of Tehran, Iran, in 1942. After working as a technical contractor for a year with the US army forces in Iran he had moved to the USA in 1944. There, he continued his studies at MIT where he received an M.S. degree in 1946. In 1949 he had obtained a position at Columbia University in New York as an instructor responsible for teaching the theories of circuits and electromagnetism.

Then, also in that year, he had the opportunity to organize and moderate a debate meeting about digital computers in which Shannon, Berkeley and Francis J. Murray (1911-1996) took part. It was probably the first public debate on this subject ever! [42]

Due to this deep interest he turned his attention to the problems of computers when he had received his Ph.D., and in 1950, when he became an assistant

professor, he published the paper on "Thinking Machines" in *The Columbia Engineering Quarterly*, an electrical engieneering students' journal [49]. He also featured such headlines:

> "'Psychologists Report Memory is Electrical', 'Electric Brain Able to Translate Foreign Languages is Being Built', 'Electronic Brain Does Research', 'Scientists Confer on electronic Brain' — these are some of the headlines that were carried in newspapers throughout the nation during the past year. What is behind the headlines? How will 'electronic brains' or 'Thinking Machines' affect our way of living? What is the role played by electrical engineers in the design of these devices? These are some of the questions that we shall try to answer in this article." ([49], p. 12.)

You would think that possible answers to these questions were given in science fiction stories that have been written very soon after World War II but it is not as simple as that! One of the most famous writers of such stories was Isaac Asimov (1919–1992) who started writing sold SF stories — of course without any computers contained — for the *Amazing Stories* magazine in the late 1930s.

In the 1940s Asimov wrote many of his robot stories and some of them, e.g. "The Bicentennial Man", were made into later films. Also other contents of Asimov's robot stories have been incorporated in the settings of SF movies in the last three decades, e.g. "Little Lost Robot" that was first published in the March 1947 issue of *Astounding Science Fiction* and "I, Robot" that was originally a SF short story by Eando Binder[1], already published in the January 1939 issue of *Amazing Stories*. This story influenced Asimov to write nine robot stories in the collection *I, Robot* [1].[2]

SF action movies of the 1990s until today, e.g. *Matrix* and *Terminator*, show scenarios of the world where computers run countries or govern the whole mankind. In *The Terminator*,[3] that plays in the year 2029 after an apocalypse, artificially intelligent machines intend to exterminate all human beings. The American SF action film series of *The Matrix*[4] describes the fight of a small group of humans against artificial intelligent machines that dominate the Earth of the 21st century. These machines control the minds of all other humans by implants connecting them to a simulated reality called "The Matrix".

It is important to notice that these dystopic scenarios of a world, that is dominated by computers, were not in the settings of that early science fiction stories. On the

[1] Under the Eando name, the brothers Earl Andrew Binder (1904–1965) and Otto Binder (1911–1974) ("E" and "O Binder") wrote science fiction stories on a robot named Adam Link.

[2] These stories originally appeared in the American magazines *Super Science Stories* and *Astounding Science Fiction* between 1940 and 1950. Asimov had titled this collection *Mind and Iron* but the publisher changed the title without his approval.

[3] *The Terminator*, 1984, *Terminator 2: Judgment Day*, 1991, and *Terminator 3: Rise of the Machines*, 2003, American science fiction action films; Director: James Cameron, Co-writers: James Cameron, William Wisher Jr., and *Terminator Salvation*, 2009 Director: Joseph McGinty Nichol, Co-writers: John Brancato and Michael Ferris.

[4] *The Matrix*, 1999, *The Matrix Reloaded*, 2003, and *The Matrix Revolutions*, 2003; Directors and Writers: Larry and Andy Wachowski.

contrary, a large number of them have been "Computer-is-God stories" wrote author John Clute in 1995 in his *Illustrated Encyclopedia* on science fiction:

Later the "Computer-is-God stories" turned into "Computer-can-think stories". An example is the movie *Wargames*[5], a Cold War story on computer-controlled nuclear disarmament. The movie begins with a simulated nuclear attack to the USA by the Soviet Union. It turns out that 22% of the US Air Force Strategic Missile Wing missileers prove unwilling to turn a key required to launch a missile strike. Therefore the command of missile silos is maintained through automation, without human intervention. Control is given to the NORAD[6] computer, WOPR (War Operation Plan Response), a learning expert system.

A high school student hacks this computer by accident when he looks for computer games. Thinking that he found a forthcoming game, he starts the program "Global Thermonuclear War" playing as the Soviet Union. It seems that this ends in a disaster, but then the protagonist suggests the computer to play Tic-Tac-Toe against itself. WOPR learns the concept of futility and concludes: "A strange game. The only winning move is not to play." Cycling through all the nuclear war scenarios it has devised, it finds that they to all result in stalemates: "WINNER: NONE" is the output on the screen and WORP cancels the launch of the second strike. In this movie the computer became a rational thinker, an artificial or at least a computational being that recompensed the fooling of human beings by making its own decision!

In 1979 and in 1984 two books appeared with almost equal titles: *Machines Who Think*, Pamela McCorduck's *Personal Inquiry Into the History and Prospects of Artificial Intelligence* [24] that is still a very good approach to the early history of AI[7], and *Machines That Think* [3], a compilation of 29 science fiction stories of the 20th century that originally have been published in the period of 1909–1973.[8] That collection (edited by Asimov et al.: [3]) comprises interesting examples of speculations of computers and robots in their respective future.

2 The Computer Era in Science Reality and Science Fiction

Traditional histories of computation, computers and Artificial Intelligence (AI) consider Alan Turing as the "father" of both computing and AI because he set up the mathematical basis for the theory of computation while still being a graduate student at Princeton University in 1936. He developed the concepts that now are considered as the basic elements of computation. His main contribution was to apply the idea of a "methodical process" (what people perform when pursuing any kind of organized action) to something that can be done "mechanically" by a machine. Though

[5] *WarGames*, 1983, American science fiction film; Director: John Badham; Co-writers: Lawrence Lasker and Walter F. Parkes.

[6] North American Aerospace Defense Command.

[7] The book appeared in 2004 in a new "25th anniversary edition" [25].

[8] The first story is Ambrose Pierce's *Moxon's Master* and the last is *Starcrossed*, written by George Zebrowski.

he didn't construct such a device, he mathematically demonstrated that this could be possible by proposing a hypothetical machine known since then as the "Turing machine". Turing gave for the first time the formal definition of what should count as a "definite method" (or, in modern language, simply "an algorithm"). That machine would be able to perform certain elementary operations by using a series of instructions, which have to be written in symbols of formal language (that is, in a *precise* form). His idea was that these symbols could be translated into a physical medium (which in Turing's example consisted on a paper tape). An "effective algorithm" was defined by Turing as a series of instructions that, applied to a set of data, allow us to achieve correct results. Turing's argument goes on by telling that, if each particular algorithm can be written out as a set of instructions in the same standard form, there could be a universal machine that can do what any other particular Turing machine would do. The "Universal Turing Machine" embodies the essential principle of the computer: a single machine for all possible tasks.

This abstract machine that represented the process of computing on a paper band subdivided into fields could solve every conceivable mathematical problem as long as there was an algorithm for it. In his paper "On Computable Numbers, with an Application to the Entscheidungsproblem" [46], Turing reformulated Kurt Gödel's (1906-1978) incompleteness-findings and he replaced Gödel's universal, arithmetic based, formal language with simple, formal "automata".

Turing's machine was a purely theoretical model, a kind of universal computer. However, this abstract idea of an automatic calculating machine was to be realized ten years later in the so-called era of computers that started in the 1940s. The first one was the electro-mechanical Z3 computer that was designed in 1941 by Konrad Zuse (1919–1995) in Berlin, Germany; the second was "the first electronic computer ABC" (Atanasoff-Berry-Computer) created by John V. Atanasoff (1903–1995) and Clifford E. Berry (1918–1963)[9]; the third were the digital and electronic "Colossus" computers in England designed by Tommy Flowers (1905–1998) with the help of Turing.[10] These were used to decrypt the Germans' Enigma codes during the Second World War in 1943. In 1944 we had the first large-scale electro-mechanical and digital computer, the Automatic Sequence Controlled Calculator (ASCC), later renamed Harvard Mark I, in 1947 Mark II, in 1949 the mostly-electronic Mark III and in 1952 the then all-electronic Mark IV. This series was conceptual designed by Howard Aiken (1900–1973), from 1944 to 1949 the mathematician Grace Murray Hopper (1906–1992) joined the project (for details see [36]).

Based on Turing's achievements, the idea of a "computing machine" changed in the late 1940s from the earlier conception of "computers" (or sometimes "computors") as women that performed computations, to apply that name to the machine that, based on digital equipment, was able to perform anything that could be described as "purely logical". Because of his demonstration that computation could be used for more than just mathematical calculations, the study of computability began to be a "science".

[9] However, the "ABC" was not general-purpose computer.

[10] "Colossus" was also a not general-purpose computer.

In 1945, almost one year before ENIAC was announced, the mathematician John von Neumann (1903–1957) was asked to prepare a report on the logical principles of its successor, the EDVAC (since the ENIAC had not had any such description and it had been sorely missed). The ENIAC had an electronic working memory, so the individual processing operations of the entered data were exceptionally fast. However, each program to be run had to be hard wired, and so reprogramming required several hours of work.

Von Neumann recognized very quickly, though, that this was a major drawback to the huge computer, and he was soon looking for ways to modify it. Today, the novel concept of a central programming unit in which programs are stored in a coded form is attributed to von Neumann. Instead of creating the program by means of the internal wiring of the machine, the program is installed directly in the machine. Basic operations like addition and subtraction remain permanently wired in the machine, but the order and combinations of these basic functions could be varied by means of instructions that were entered into the computer just like the data. The EDVAC was not supposed to suffer from the "childhood diseases" that had afflicted the ENIAC. To this end Neumann's principle of store programming was used and the principle that went down in the history of the computer as "Von Neumann architecture" was realized for the first time [31]. This "Von Neumann architecture" became also the basis of the first computer built under his direction by the Institute for Advanced Study (IAS) in Princeton, New Jersey.

In 1949, the year before *Eckert-Mauchly Computer Corporation* (EMCC) was sold to Remington Rand, Grace Hopper had become senior mathematician and joined the team developing the UNIVAC I and she developed the idea of machine-independent programming languages. Then, in March 1950, the UNIVAC computer (UNIVersal Automatic Computer) was delivered. This machine not only became known as the first commercial computer but also for predicting the outcome of the U.S. presidential election in the following year.

2.1 The "Absurdity of Computer Science Fiction"

"History is all about what already happened. So the historian and the science fiction writer might seem to be the two heads of Janus. One stares at the past, and tries to imagine how it might have been different and why it wasn't. The other stares at the future, and wonders how it will be." [14] Following the argument of the historian of computer technology, Thomas Haigh along with the American science fiction writer Kim S. Robinson, we can see here that "science fiction is 'an historical literature'." [14] Moreover, Robinson notes that in any work of science fiction "there is an explicit or implicit fictional history that connects the period depicted to our present moment or to some moment of our past" and Haigh concludes in Robinson's words that science fiction and historical fiction "are more alike, in some respects, than either is like the literary mainstream … both are concerned with alien cultures, and with estrangement." ([14], p. 8, [35])

Haigh names it the "absurdity of science fiction as a literature of prediction, and its merit as a genre of historical writing" that "can be seen particularly clear in its treatment of computing. Computers show up in science fiction in the early 1950s, mirroring their arrival in the real world." ([14], p. 9.)

The computing machines constructed in the first two decades of the computer era looked different from our 21st century imaginations of a computer, they acted different and they were different. As Clute wrote in 1995, "they were great, clumsy giants, and hardly more powerful than a wristwatch is today." That book was published a long time ago and today's wristwatches get ready to become or to merge with today's computers. But the essence of Clute's argument is true and he continued describing these machines of the 1940s and 1950s as follows: "The basic technology available still functioned, very crudely, through enormous and unreliable gadgets like vacuum tubes, programmed via hand-punched punch cards. It was a nightmare in the real world but it was not a nightmare in the world of SF." ([10], p. 74)

The scientists who built the first computers used them for scientific calculation and in these first years, SF writers paid almost no attention to them. ([10], p. 74) "At first glance this is strange." Clute wrote then, and he referred to the fact that many of the SF authors had a background or even a degree in a scientific or engineering discipline. For instance, Asimov[11] joined in 1951 the faculty of the Boston University School of Medicine but in 1958 he resigned from this position to become a full-time writer. Could SF writers with such a scientific background miss this technological revolution? — Clute plausibly surmises that "the computer appeared to present a challenge to *Homo sapiens*. The small amount of speculation about computers that appeared before the 1960s failed to see them as almost infinitely adaptable tools, concentrating instead on visions in which computers replaced humanity, or took over from humanity, or became God. The computer was not imagined simply because to do so was to welcome into our bosoms the ultimate enemy." ([10], p. 75)

Haigh indicates that "Computers were unknown in Asimov's best-known work of this era, the *Foundation Trilogy* (originally published from 1942 to 1950). Fifty thousand years from now scientists have achieved some miracles of miniaturization, including shrinking nuclear reactors to the size of walnuts for use in atomic-powered dishwashers and personal force fields. But they don't seem to have invented computers. A separate stream of stories explored the three laws of robotics, depicting the development of ever more intelligent and human-like machines powered by the rather nebulous technology of "positronic brains". Robots are common but computers remain very rare; a handful of "thinking machines" with "super robot brains" are used for economic control and scientific research. Asimov also wrote, from 1955 onward, a handful of stories concerned with a giant computer named Multivac, built with vacuum tubes and buried deep underground. This machine too fits the "giant brain" paradigm, and comes eventually to rule the world." ([14], p. 10.)

[11] Asimov got a Bachelor in 1939 and a Master's degree in 1941 from Columbia University in New York, and after the War in 1948 he returned to Columbia University to earn a Ph.D. in biochemistry.

Fig. 1 "Thinking Machine"; illustration in Eando Binder's story "The Cosmic Blinker", art by Frank R. Paul, 1953

2.2 Asimov's MULTIVAC

MULTIVAC is the name of a fictional supercomputer in some of Asimov's stories in the 1950s and — almost needless to say — it is an allusion to UNIVAC. Initially the name should mean "MULTIple VACuum tubes" but in 1956 in the story "The Last Question" Asimov translated the suffix AC to be "Analog Computer". In all the stories MULTIVAC is a computer that operates in ordinary to the government for security purposes. In the various stories MULTIVAC has different skills.

For one, in *Franchise* (1955) the future United States of the year 2008 use the system of a so-called "electronic democracy". A single person has been selected by the computer MULTIVAC to answer some questions. Then, MULTIVAC uses the

answers and other data to predict the results of an election. Therefore, no actual election will be held.[12]

Other MULTIVAC-stories are: *Question* (1955); *Jokester* (1956); *The Last Question* (1956); *All the Troubles of the World* (1958); *The Machine that Won the War* (1961); *The Life and Times of MULTIVAC* (1975); *Point of View* (1975).

When Asimov published the story *Profession* in 1957, he told the reader of a society on the planet Earth in the 65th century. In that far future children are taught to read at the age of eight to eighteen and after that they will be educated by a process that is called "taping", i.e. a brain-computer interface. A computer analyzes the brain of every child and this analysis is the basis to determine their future profession at their "Education Day". The young humans have no chance to object or resist and the best educated of them have to compete in their profession in futuristic Olympic games. The winners in these "Olympics" have the chance of being "bought by an advanced Outworld if they are valuable for the colonies whereas to stay on Earth means to have an inferior status.

2.3 Zadeh's "Thinking Machines"

It seems that it originated from the early Computer Science and that it was Lotfi A. Zadeh who published the first science vision of a "Thinking Machine" in 1950 and he also predicted then that "Thinking Machines" may be commonplace anywhere from ten to twenty years hence and that they will play a major role in any armed conflict. Not until decades later new SF stories told these kinds of stories of computers that govern the world by their own decision — sometimes they annihilate the earth, sometimes they protect the planet.

In 1950, when Zadeh wrote "Thinking Machines: A New Field in Electrical Engineering" [49] (Figure 2), he was interested in "the principles and organization of machines which behave like a human brain, and as we said already, such machines were then variously referred to as "thinking machines", "electronic brains", "thinking robots", and similar names and with this article, he wanted first to clarify how a thinking machine differed from other machines. To do so, he used a very simple example:

> However, an idea of the principles involved in a thinking machine can be obtained from the description of a Tit-Tat-Toe playing device which was recently demonstrated by Robert Haufe at Caltech before a meeting of the American Institute of Electrical Engineers. ([49], p. 12)

In addition to chess, Tit-Tat-Toe, today better known as Tic-Tac-Toe, was one of the games that scientists wanted to teach machines to play early on. The game is played by two players on a board of three by three squares. The players take turns filling a square with their respective symbols (usually $\times$ and $\circ$). The game is won by the player who manages to place three of his symbols in a row (horizontally, vertically

[12] Contingently the correct predicting of the U.S. presidential election's outcome in the year 1952 motivated Asivmov to write this story.

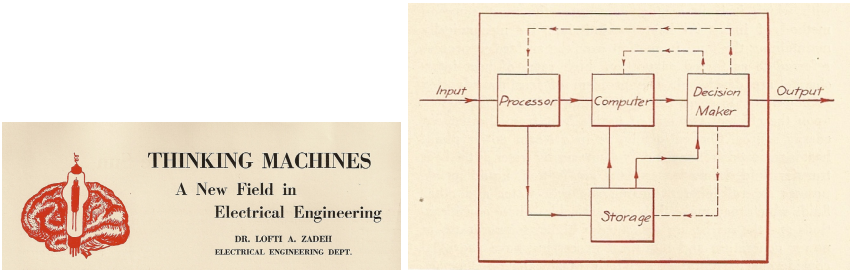


Fig. 2 Left: Illustration accompanying Zadeh's article [49], the author's first name was misspelled here. Right: Zadeh's chart for the basic elements of a "Thinking Machine"

or diagonally). Haufe's machine functioned with relay circuits. It saved information about which of the individual fields were filled with the players' symbols, it could make sensible moves and could indicate the result at the end of the game. When it was the machine's turn, it classified all nine fields according to whether or not filling them was strategically desirable. These classes were then searched for empty fields. An empty field with the highest strategic value was then filled. ([15], p. 885.)

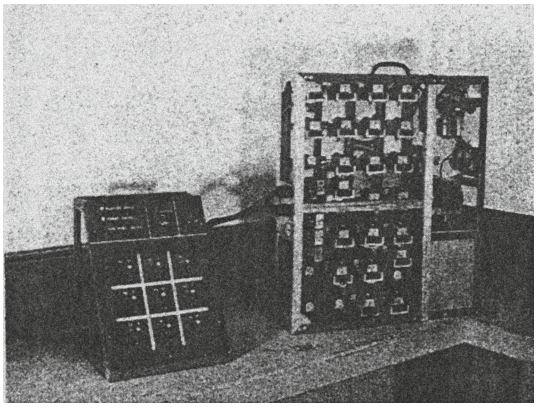


Fig. 3 The two units comprising Robert Haufe's Tit-Tat-Toe machine

Haufe's Tic-Tac-Toe machine, which Zadeh displayed in his article (Figure 3), was naturally much simpler than other machines that were referred to as "thinking machines". However, Zadeh considered the ability to make decisions to be a characteristic feature of thinking machines:

> Despite its simplicity, Haufe's machine is typical in that is possesses a means for arriving at a logical decision based on evaluation of a number of alternatives. More generally, it can be said that a thinking machine is a device which arrives at a certain decision or answer through a process of evaluation and selection. ([49], p. 13)

Zadeh's diagram (Fig. 2) demonstrates the "thinking" process of such machines: Incoming input data are sorted and processed in the processor. Some of this processed data is then sent to storage to be saved for later use (this storage can be in the form of punch cards, tapes or cathode ray tubes), "and it has the same function as memory in a human brain". ([49], p. 13)

Another portion of the processed data as well as some of the saved data are called up into the unit known as the computer where necessary calculations are performed. The computer is not the essential component of the thinking machine, however, unless the calculation either is the end result itself or will be needed at the end in order to make the decision. More important is the *decision maker*, for it is here that decisions are reached. All of the relevant information coming from the computer and from storage is evaluated and weighted according to the commands and criteria present within the machine. The final answer or decision is formed on this basis as output. Dashed lines lead from the decision maker to all three elements of the machine: These are the so-called feedback connections. This feedback allows the three elements to operate as a function of the data obtained from the decision maker as needed.

In a footnote Zadeh mentioned that the "same names are frequently ascribed to devices which are not 'thinking machines' in the sense used in this article", therefore he separated them as follows: "The distinguishing characteristic of thinking machines is the ability to make logical decisions and to follow these, if necessary, by executive action." ([49], p. 12.)

He stated: "More generally, it can be said, that a thinking machine is a device which arrives at a certain decision or answer through the process of evaluation and selection." With this definition he decided: "Thus, M.I.T.'s differential analyser is not a thinking machine, for it can not make any decisions, except trivial ones, on its own initiative. However, the recently built large-scale digital computers, UNIVAC and BINAC[13], are endowed with the ability to make certain non-trivial decisions and hence can be classified as thinking machines." ([49], p. 13.) Zadeh explained in this article "how a thinking machine works" and he claimed that "the box labeled Decision Maker is the most important part of the thinking machine".

2.4 Zadeh's "Electronic Admission Director"

Zadeh found a very interesting exemplification of his imagination of a "Thinking Machine" in 1950 and this picture resembles the concept of Asimov's computer in *Profession*. He illustrated his argumentation by peering forward into the year 1965. Three years earlier, in this version of the future, the administration at Columbia University had decided, for reasons of economy and efficiency, to close the admissions office and install in its place a thinking machine called the "Electronic Admissions Director". The construction and design of this machine had been entrusted to the electrical engineering department, which completed the installation in 1964.

[13] The Binary Automatic Computer was the first stored-program computer in the US, built at EMCC in 1949.

Since then, the "director" has been functioning perfectly and enjoying the unqualified support of the administration, departments and students. This thinking machine functions as follows:

1. Human secretaries convert the information from the list of applicants into series of numbers $a_1, a_2, a_3, \ldots, a_n$; each number represents a characteristic, e.g. a_1 could stand for the applicant's IQ, a_2 for personal character, and so on.
2. The lists coded thusly are provided to the *processor*, which processes them and then relays some of the data to the *computer* and another part of the data to *storage*. On the basis of applicant data as well as university data, the *computer* calculates the probabilities of various events, such as the probability that a student will fail after the first five years. This information and the saved data are sent to the *decision maker* to come to final decision on whether to accept the applicant. The decision is then made based on directives, such as these two:

   - accept if the probability of earning the Bachelor's degree is greater than 60%;
   - reject if the probability that the applicant will not pass the first year of college is greater than 20%.

Zadeh didn't consider the machine sketched out here to be as fanciful as student readers (and surely others, as well) may have thought: Machines such as this could be commonplace in 10 or 20 years and it is already absolutely certain that thinking machines will play an important role in armed conflicts that may arise in the future. ([49], p. 30) Back then, in the year 1950, though, there was still much to be done so that these or similar scenarios of the future could become reality.

3 Making Computers Think (Like People?)

In 1948 the young mathematician John McCarthy (1927–2011) had attended the "Hixon Symposium on Cerebral Mechanisms in Behaviour" at Caltech where he became acquainted with Warren S. McCulloch (1898–1969) who gave his famous talk on "Why the Mind Is in the Head" [39], and other well-known members of the so-called "Cybernetics group" [16]. During that symposium he also met the mathematicians Turing and von Neumann, the psychologist Karl S. Lashley (1890–1958) and Claude Shannon. This event initiated his life-time interests related to the development of "machines that could think". In the following year he changed to Princeton to study automata models with von Neumann and he became friends with his fellow student Marvin L. Minsky (born in 1927). In 1951 he received his Ph.D. graduation at Princeton University, he spent the following year at Bell Labs and he eventually discussed the idea of machine intelligence with Shannon. He argued him into collecting and publishing scientific works on machines that seem to be intelligent. The two edited the well-regarded and influential collection *Automata Studies* that got this technical title because Shannon did not like provoking headings. The voluminous tome appeared in print in 1956 [45] and here Automata theory and Turing machines were treated from different sides. But McCarthy was dissatisfied with the content of these papers concerning the potential of creating "intelligent

computers"! In the mid-1950s he wished "to nail the flag to the mast," he said at the AI@50-conference in 2006 [21].

3.1 Artificial Intelligence

In 1955 McCarthy got an assistant professorship of mathematics at Dartmouth College in Hanover, New Hampshire. Here he initiated AI when he started to organize a "Summer Research Project on Artificial Intelligence" modeled on the traditional military summer schools. In the proposal to this project McCarthy wrote that AI-research will "proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it" [23]. Even this provoking text and moreover the provoking name "Artificial Intelligence" in the heading that were McCarthy's conceptions, the proposal appeared officially under the authorship of McCarthy, Minsky, the prominent IBM computer designer Nathaniel Rochester (1919–2001) and the well-known Shannon who was in that year going to join MIT.

The Dartmouth workshop was held during one month in the summer of 1956 and ten people took part, among the organizers McCarthy, Minsky, Rochester and Shannon, there were Raymond J. Solomonoff (1926-2009), Oliver G. Selfridge (1926-2008), Trenchard More, Arthur L. Samuel (1901-1990), Herbert A. Simon (1916-2001) and Allan Newell (1927-1992). It was a meeting of brainstorming discussions on the potential of information technology between experts in language, sensory input, learning machines and other fields; it helped focusing AI research for the future. McCarthy recalled later that he was "disappointed in how few research papers dealt with making machines behave intelligently. [...] But the real reason we didn't live up to grand hopes was that AI was harder than we thought." [21] James Moor wrote what McCarthy also emphasized basically when he spoke at the AI@50-meeting: "Nevertheless there were important research developments at the time, particularly Allen Newell's, John C. Shaw's (1922–1991), and Herbert Simon's Information Processing Language (IPL) and the Logic Theory Machine" ([29], p. 87). The system of this Carnegie Mellon-researcher trio was proving elementary logical theorems and playing games. Symbols for objects like chess figures or truth values had to be manipulated by the used program language and to this end the three established the concept of "list structures" that enthused the other participants. Minsky tried to build a geometry problem solver as an application of the rule-based approach that was proposed by Newell and Simon and Rochester and Herbert Gelernter started the trial to implement the program. Moor summed up that the Dartmouth project "was not really a conference in the usual sense. There was no agreement on a general theory of the field and in particular on a general theory of learning. The field of AI was launched not by agreement on methodology or choice of problems of general theory, but by the shared vision that computers can be made to perform intelligent tasks" ([29] p. 87).

The subsequent history of AI research is a story of several successes but has yet lagged behind expectations. AI became a field of research to build computers

and computer programs that act "intelligently" although no human being controls those systems. AI methods became methods to compute with numbers and find exact solutions. However, not all problems can be resolved with these methods. On the other hand, humans are able to resolve such tasks very well, as Zadeh mentioned in many speeches and articles over the last decades. In conclusion, he stated that "thinking machines" do not think as humans do. From the mid-1980s he focused on "Making Computers Think like People" [52]. For this purpose, the machine's ability "to compute with numbers" should be supplemented by an additional ability that is similar to human thinking: Computing with Words and Perceptions. To this end a new mathematical theory was necessary.

Fig. 4 Zadeh moderating an annual AI debate at Berkeley; from left to right: John McCarthy, Lotfi A. Zadeh, Hubert Dreyfus (University of California, Berkeley)

3.2 Fuzzy Sets and Systems

In 1959 Zadeh became professor at the University of California at Berkeley and in the course of writing the book *Linear System Theory: The State Space Approach* [56] with his colleague Charles A. Desoer (1926–2010), he "began to feel

that complex systems cannot be dealt with effectively by the use of conventional approaches largely because the description languages based on classical mathematics are not sufficiently expressive to serve as a means of characterization of input-output relations in an environment of imprecision, uncertainty and incompleteness of information." [41] There were two ways to overcome this situation. In order to describe the actual systems appropriately, he could try to increase the mathematical precision even further, but Zadeh failed with this course of action. The other way presented itself to Zadeh in the year 1964, when he discovered how he could describe real systems as they appeared to people. "I'm always sort of gravitated toward something that would be closer to the real world" [42].

In order to provide a mathematically exact expression of experimental research with real systems, it was necessary to employ meticulous case differentiations, differentiated terminology and definitions that were adapted to the actual circumstances, things for which the language normally used in mathematics could not account. The circumstances observed in reality could no longer simply be described using the available mathematical means.

While he was serving as Chair of the department in 1963/64, he continued to do a lot of thinking about basic issues in systems analysis, especially the issue of unsharpness of class boundaries. These thoughts indicate the beginning of the genesis of Fuzzy Set Theory." ([55], p. 7).

In his first article "Fuzzy Sets" he launched new mathematical entities as classes or sets that "are not classes or sets in the usual sense of these terms, since they do not dichotomize all objects into those that belong to the class and those that do not." He introduced "the concept of a fuzzy set, that is a class in which there may be a continuous infinity of grades of membership, with the grade of membership of an object x in a fuzzy set A represented by a number $f_A(x)$ in the interval $[0, 1]$." [50][14]

Since that time he often compared the strategies of problem solving by computers on the one hand and by humans on the other hand. In a conference paper in 1970 he called it a paradox that the human brain is always solving problems by manipulating "fuzzy concepts" and "multidimensional fuzzy sensory inputs" whereas "the computing power of the most powerful, the most sophisticated digital computer in existence" is not able to do this. Therefore, he stated that "in many instances, the solution to a problem need not be exact", so that a considerable measure of fuzziness in its formulation and results may be tolerable. The human brain is designed to take advantage of this tolerance for imprecision whereas a digital computer, with its need for precise data and instructions, is not." ([51], p. 132) He continued: "Although present-day computers are not designed to accept fuzzy data or execute fuzzy instructions, they can be programmed to do so indirectly by treating a fuzzy set as a data-type which can be encoded as an array [...]." Granted that this is not a fully satisfactory approach to the endowment of a computer with an ability to manipulate fuzzy concepts, it is at least a step in the direction of enhancing the ability of machines to emulate human thought processes. It is quite possible, however, that truly significant advances in artificial intelligence will have to await the

[14] For more details on the genesis of the theory of Fuzzy Sets and Systems see: [43], chapter V.

development of machines that can reason in fuzzy and non-quantitative terms in much the same manner as a human being." ([51], p. 132)

3.3 Artificial Neural Networks

In 1943 Warren McCulloch and the 19-year-old math student Walter H. Pitts (1923–1969) published "A Logical Calculus of the Ideas Immanent in Nervous Activity" [26]. The text linked the activities of a network of abstract electric on-off switches, so-called neurons, with a complete logical calculus for time-dependent signals in electric circuits with synaptic delays. By modeling these neurons after electric on-off switches, which can be interconnected such that each Boolean statement can be realized, McCulloch and Pitts now "realized" the entire logical calculus of propositions by "neuron nets".

Every McCulloch-Pitts neuron is a threshold element: If the threshold value is exceeded, the neuron becomes active and "fires". By "firing" or "not firing", each neuron represents the logical truth values "true" or "false". Appropriately linked neurons thus carry out the logical operations like conjunction, disjunction, etc.

Two years later von Neumann picked the paper up and used it in teaching the theory of computing machines [31] and may be that initiated the research program of "Neuronal Information Processing", a collaboration involving psychology and sensory physiology, in which other groups of researchers were soon interested. Some years later, von Neumann wrote on his comparative view on the computer and the brain in an unfinished manuscript that was published posthumously after he died because of cancer.[30]

In 1951, Minsky had worked with Dean Edmonds in Princeton to develop a first neurocomputer, which consisted of 3,000 tubes and 40 artificial "neurons" was called SNARC (Stochastic Neural-Analog Reinforcement Computer), in which the weights of neuronal connections could be varied automatically. But SNARC was never practically employed.

The classic problem that a computer at that time was supposed to solve, and hence an artificial neuronal network was expected to, was the classification of patterns of features, such as handwritten characters. Under the concept of a pattern, objects of reality are usually represented by pixels; frequency patterns that represent a linguistic sign, a sound, can also be characterized as patterns. In 1957/1958, Frank Rosenblatt (1928–1971) and Charles Wightman at Cornell University developed a first machine for pattern classification. Rosenblatt described this early artificial neuronal network, called Mark I Perceptron, in an essay for the *Psychological Review* [37]. It was the first model of a neuronal network which was capable of learning and in which it could be shown that the proposed learning algorithm was always successful when the problem had a solution at all.

The perceptron appeared to be a universal machine and Rosenblatt had also heralded it as such in his 1961 book *Principles of Neurodynamics: Perceptrons and the Theory of Mind*:

> "For the first time, we have a machine which is capable of having original ideas. ... As concept, it would seem that the perceptron has established, beyond doubt, the feasibility and principle of nonhuman systems which may embody human cognitive functions ... The future of information processing devices which operate on statistical, rather than logical, principles seems to be clearly indicated." [37]

The euphoria came to an abrupt halt in 1969, however, when Minsky and Seymour Papert (born 1928) completed their study of perceptron networks and published their findings in a book. [27] The results of the mathematical analysis to which they had subjected Rosenblatt's perceptron were devastating: Artificial neuronal networks like those in Rosenblatt's perceptron are not able to overcome many different problems! For example, it could not discern whether the pattern presented to it represented a single object or a number of intertwined but unrelated objects. The perceptron could not even determine whether the number of pattern components was odd or even. Yet should this have been a simple classification task that was known as a "parity problem". The either-or operator of propositional logic, the so-called XOR, presents a special case of the parity problem that thus cannot be solved by Rosenblatt's perceptron. Therefore, the logical calculus realized by this type of neuronal networks was incomplete. As a result of this fundamental criticism, many projects on perceptron networks or similar systems all over the world were shelved or at least modified. It took many years for a revival of this branch of AI research.

Since 1981 the psychologists James L. McClelland (born in 1948) and David E. Rumelhart (1942–2011) applied Artificial Neural Networks to explain cognitive phenomena (spoken and visual word recognition). In 1986, this research group published the two volumes of the book *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* [38]. Already in 1982 John J. Hopfield, a biologist and Professor of Physics at Princeton, CalTech, published the paper "Neural networks and physical systems with emergent collective computational abilities" [18] on his invention of an associative neural network (now more commonly known as the "Hopfield Network"), i.e.: Feedback Networks that have only one layer that is both input as well as output layer and each of the binary McCulloch-Pitts Neurons is linked with every other, except itself.

McClelland's research group could show that perceptrons with more than one layer can realize the logical calculus; multi layer perceptrons were the beginning of the new direction in AI: Parallel Distributed Processing.

3.4 Evolutionary Strategies

In 1940 the immigrated mathematician Stanislaw Ulam (1909–1984) had studied the growth of crystals at Los Alamos National Laboratory. When looking for a model of discrete dynamic systems he had the idea of a cellular automaton and he created a simple lattice network. The states of Ulam's cells at a certain point in time t were determined by its state at point in time instantaneous before t. Von Neumann picked up this idea in 1953 when he conceptualized a theory of self-reproducing two-dimensional cellular automata with a self-replicator implemented algorithmic.

There was a universal copier and constructor working within a cellular automaton with 29 states per cell and von Neumann could show that a particular pattern would copy itself again and again within the given pool of cells.

Following up this concept von Neumann was wondering if self-reproducing of automata also could pursue an evolutionary strategy, i. e. due to mutations and struggle for resources. Unfortunately there is no paper by von Neumann on this subject. Arthur W. Burks (1915–2008), a mathematician and philosopher who was von Neumann's collaborator in the IAS computer project since 1946, expanded the theory of automata, completed and edited the paper "Theory of Self-Reproducing Automata", von Neumann had been working on, posthumously. The paper was published in 1966 and it "had a huge impact, not only in computing but in biology and philosophy as well," said John H. Holland (born 1929), professor of psychology, electrical engineering and computer science at the University of Michigan in Ann Arbor. "Until then, it was assumed that only living things could reproduce." [19] and shortly after this field of research was named "Evolutionary Computing". Holland was a member of Burk's "Logic of Computers Group", in 1954 he was among the first students in the new Ph.D. program "Computer and Communication science" and the first to graduate in 1959.

Holland was affected by the book *The Genetical Theory of Natural Selection*, written by English statistician and evolutionary biologist Sir Ronald A. Fisher (1890–1962), and he was warm on analogies of evolutionary theory and animal breeding from a computer science point of view: Can we breed computer programs? "That's where genetic algorithms came from. I began to wonder if you could breed programs the way people would say, breed good horses and breed good corn", Holland recalled later ([28], p. 128). In his *Adaptation in Natural and Artificial Systems*, that he published in 1975, he showed how to use these "genetic" search algorithms to solve real-world problems. His research objectives were i) the theoretical explanation of adaptive processes in nature and ii) the development of software that keeps the "mechanisms" of natural systems and adapting to the respective circumstances at the best. [17]

The name "Genetic Algorithms" goes back to the Ph.D. thesis of John D. Bagleys under Holland's supervision [4]. Bagley applied these algorithms to find solutions in game theoretic problems and more of Holland's Ph. D students, e.g. Kenneth De Jong and David E. Goldberg, could demonstrate other successful applications.

Almost at the same time so-called "Evolutionary Programming" appeared with the research work of Lawrence G. Fogel (1928–2007) from the University of California, Los Angeles [13]. In 1966 the book *Artificial Intelligence through Simulated Evolution* appeared, co-authored by Fogel, Al Owens und Jack Walsh. These biological inspired research programs merged to the now so-called field of "Evolutionary Computation."

Apart from these developments in the US other natural inspired principles have been considered in Germany: In the 1960s, Ingo Rechenberg (born 1934) and Hans-Paul Schwefel (born 1940), two students of aircraft construction at the Technical University of Berlin, suggested to consider the theory of biological evolution to develop optimization strategies in engineering. In 1963 they founded the

"inofficial working group Evolutionstechnik" performing "experimental optimization" of the shape of wings and kinked plates through mostly small modifications of the variables via a random manner. This was the seminal idea of the research field "Evolution Strategies", which was initially handled without computers. However, some time later, Schwefel expanded the idea toward evolution strategies to deal with numerical/parametric optimization and, also, formalized it as it is known nowadays. [34], [40].

4 Soft Computing

"The concept of soft computing crystallized in my mind during the waning months of 1990" wrote Lotfi Zadeh in 2001 [53]. He coined this label 'Soft Computing' (SC) to name an interdisciplinary field that covers different approaches to Artificial Intelligence that had been developed during the last decades but weren't part of the mainstream of AI: He formulated this new scientific concept when he wrote that

> "what might be referred to as soft computing — and, in particular, fuzzy logic — to mimic the ability of the human mind to effectively employ modes of reasoning that are approximate rather than exact. In traditional — hard — computing, the prime desiderata are precision, certainty, and rigor. By contrast, the point of departure in soft computing is the thesis that precision and certainty carry a cost and that computation, reasoning, and decision making should exploit — wherever possible — the tolerance for imprecision and uncertainty. [...] Somewhat later, neural network techniques combined with fuzzy logic began to be employed in a wide variety of consumer products, endowing such products with the capability to adapt and learn from experience. [...] Underlying this evolution was an acceleration in the employment of soft computing — and especially fuzzy logic — in the conception and design of intelligent systems that can exploit the tolerance for imprecision and uncertainty, learn from experience, and adapt to changes in the operation conditions." [52]

Zadeh defined a new approach and also a "new direction in AI" [54], because he is committed to the assumption that traditional AI couldn't cope with the future challenges. He directed his critique to the general approach of Computer Science and Engineering, which he calls "hard computing".

In the foreword to the new journal *Applied Soft Computing* he recommended that instead of "an element of competition" between the complementary methodologies of SC "the coalition that has to be formed has to be much wider: it has to bridge the gap between the different communities in various fields of science and technology and it has to bridge the gap between science and humanities and social sciences! SC is a suitable candidate to meet these demands because it opens the fields to the humanities. [...] Initially, acceptance of the concept of soft computing was slow in coming. Within the past few years, however, soft computing began to grow rapidly in visibility and importance, especially in the realm of applications which are related to the conception, design and utilization of information/intelligent systems. This is the backdrop against which the publication of *Applied Soft Computing* should be viewed. By design, soft computing is pluralistic in nature in the sense that it is a coalition of methodologies which are drawn together by a quest for

accommodation with the pervasive imprecision of the real world. At this juncture, the principal members of the coalition are fuzzy logic, neuro-computing, evolutionary computing, probabilistic computing, chaotic computing and machine learning." ([53], p. 1–2)

In 2010 Luis Magdalena, General Director of the *European Centre for Soft Computing* in Mieres, Asturias (Spain), that was founded in 2006, accompanied Zadeh in distinguishing between "Soft Computing as opposite to Hard Computing" (HC) saying that the "conventional approaches" of HC "gain a precision that in many applications is not really needed or, at least, can be relaxed without a significant effect on the solution" and that the "more economical, less complex and more feasible solutions" of SC are sufficient. He pointed out that using sub-optimal solutions "that are enough" is "softening the goal of optimization" to be satisfied with inferring "an implicit model from the problem specification and the available data." Inversely we can say that without an explicit model we will never find the optimal solution. But this is not a handicap! — SC makes a virtue out of necessity because it is a "combination of emerging problem-solving technologies" for real-world problems and this means that we have only "empirical prior knowledge and input-output data representing instances of the system's behavior."[20]

Also computer scientist Piero Bonissone stated, in these cases of "ill-defined systems", that are "difficult to model and with large-scale solution spaces" "precise models are impractical, too expensive, or non-existent. [. . .] Therefore, we need approximate reasoning systems capable of handling such imperfect information. Soft Computing technologies provide us with a set of flexible computing tools to perform these approximate reasoning and search tasks." [8]

When Hans-Jürgen Zimmermann, founding editor of the journal *Fuzzy Sets and Systems*, foresaw that the development of "hybrid systems" of "fuzzy-neuro-evo-combinations" would continue in the future, he deliberated about a name for the common field of research, which would then also become the subtitle of the journal. "Soft computing, biological computing and computational intelligence have been suggested so far." These concepts seemed to be attractive in different ways and also varied with respect to their expressive power. He suggested calling the field "soft computing and intelligence" since the other concepts seemed to place too much emphasis on "computing" "which is certainly not appropriate at least for certain areas of fuzzy set theory." [57]. Thus since the first issue of 1995 *Fuzzy Sets and Systems* has appeared with the subtitle *International Journal for Soft Computing and Intelligence*.

5 Computational Intelligence

The name "Computational Intelligence" (CI) originates from a Canadian journal on the topic of AI.[15] When this journal was founded in 1985, the editorial board chose this name "to reflect the fact that AI is distinct from other studies of intelligence in its emphasis on computational models," the editors recalled about 10 years later and

[15] `http://eu.wiley.com/WileyCDA/WileyTitle/productCd-COIN.html`

Fig. 5 In the year 2006 the *European Centre for Soft Computing* was launched. First row (among others: Lotfi A. Zadeh, and Enric Trillas; back row (among others): Rudolf Kruse, Luis Magdalena, Henri Prade, Janusz Kacprzyk.

they continued: "The name was also short enough to be catchy but general enough to reflect our purpose and attract submissions from all areas of AI." [9].

In "CI" the adjective "computational" was intended to refer to subsymbolic problem representation, knowledge aggregation and information processing. Here, we reach the basics of natural intelligence but — as a matter of course — it is important to distinguish between natural (biological) intelligence and AI.

As computer scientist Włodzisław Duch wrote in 2004, CI "is used as a name to cover many existing branches of science. This name is used sometimes to replace artificial intelligence, both by book authors and some journals." [11][16]

As Zadeh did when he launched SC, Duch directed his critique to Symbolic AI: He surmises that "the idea that all intelligence comes from symbol manipulation has been perhaps misunderstood by the AI community". He stressed that psychologists Newell, Simon and Shaw[17] of the Carnegie-Rand group[18] dealt with formal symbol manipulations when they presented the *Logical Theory Machine*, that could proof

[16] Duch referred to [33] and the above mentioned journal.

[17] They were the so-called "NSS-group", "NSS" was the name of a chess program, the initials of its authors.

[18] Carnegie-Mellon University and Rand-Corporation.

mathematical theorems in elementary logic,[19] and also three years later when they presented the *General Problem Solver* [32].

Duch pointed that these AI pioneers "wrote about physical symbols, not about symbolic variables. Physical symbols are better represented as multi-dimensional patterns representing states of various brain areas. Symbolic models of brain processes certainly do not offer accurate approximations for vision, control or any other problem that is described by continuous rather then symbolic variables. Approximations to brain processes should be done at a proper level to obtain similar functions. Symbolic dynamics [...] and extraction of finite state automata from recurrent networks [...] may provide useful information on dynamical systems, and may be useful in modelling transition between low-to-high-level processes."[12]

Moreover, in 2007 Duch noticed that the problems that "are at present solved in a best way by the AI community using methods based on search, symbolic knowledge representation, reasoning with frame-based expert systems, machine learning in symbolic domains, logics and linguistic methods", are "non-algorithmizable problems involving systematic thinking, reasoning, complex representation of knowledge, episodic memory, planning, understanding of symbolic knowledge". [12]

In early years CI was a collection of methods but now there exist attempts to characterize this research area explicitly as defined: "CI studies problems for which there are no effective algorithms, either because it is not possible to formulate them or because they are NP-hard and thus not effective in real life applications!" [12] As opposed to artificial systems, animate systems like living brains are able to solve problems for which there are no effective algorithms: "extracting meaning from perception, understanding language, solving ill-defined computational vision problems thanks to evolutionary adaption of the brain to the environment, survival in a hostile environment." [12] Accordingly: "A good part of CI research is concerned with low-level cognitive functions: perception, object recognition, signal analysis, discovery of structures in data, simple associations and control. Methods developed for this type of problems include supervised and unsupervised learning by adaptive systems, and they encompass not only neural, fuzzy and evolutionary approaches but also probabilistic and statistical approaches, such as Bayesian networks or kernel methods." Duch also recollects that "These methods are used to solve the same type of problems in various fields such as pattern recognition, signal processing, classification and regression, data mining."[12]

Also Magdalena, expressed "the idea of CI being the branch of science considering those problems for which there is not an exact model, plus those cases where the model exists but its consideration is not computationally effective, i.e., when we need to reduce the granularity or soften the goal." He also brought out that these ideas describe also "SC as the opposite to hard computing or based on its essential properties. So, apparently there is no significant difference between Soft Computing and Computational Intelligence." [20]

However, there is "little overlap between problems solved using low and high-level mental functions, although they belong to the same broader category of

[19] They showed it on the 1956 founding AI workshop in Dartmouth.

non-algorithmizable problems," Duch said and therefore he accentuates distinctly: "AI is a part of CI focusing on problems that require higher cognition and at present are easier to solve using symbolic knowledge representation. It is possible that other CI methods will also find applications to these problems in future. The main overlap areas between low and high-level cognitive functions are in sequence learning, reinforcement and associative learning, and distributed multi-agent systems. All tasks that require reasoning based on perceptions, such as robotics, automatic car driving, autonomous systems, require methods for solving both low and high-level cognitive problems and thus are a natural meeting ground for AI experts with the rest of the CI community."[12]

Another view on CI arrives at a different relationship of AI and CI; this view emerged from James Bezdek's reflections "On the Relationship between Neural Networks, Pattern Recognition and Intelligence" in 1992 [6] that let him to the first definition of CI. Bezdek considered three levels of system complexity that he named the "ABCs of neural networks, pattern recognition, and intelligence." The ABCs if interest to us are the following:

" A Artificial Nonbiological (manmade)
B Biological Physical + chemical + (??) = organic
C Computational Mathematics + manmade machines"

Bezdek illustrated his view of the relationships between these ABCs and neural nets (NN), pattern recognition (PR), and intelligence (I) in Fig. 6. Here "complexity increases from left to right and from bottom to top" and: "Familiar terms in Fig. 6 include ANN, AI, and the three biological notions in the first row."[20]

He discussed Fig. 6 starting at the uppermost row: "The BNN is one of the physiological systems that facilitates organisms (in particular, humans) to perform various biological recognition tasks. One key input to the BNN is sensory data; another 'knowledge.' In turn BPR is but one aspect of biological intelligence. Some writers refer to the BNN as the hardware of the human body, the brain; BI then corresponds to the software of the human body, the mind. At the other end of the complexity spectrum, and I believe, in an entirely analogous way, computational NNs that depend solely on sensor data are (but one!) facilitator of computational PR, which in turn is but one aspect of computational intelligence. The middle row (A = Artificial) is perhaps the most interesting, for it offers us a means of extending low-level computational algorithms upwards toward their biological inspirations." Considering "other differences . . . between the B, A, and C levels of complexity", he emphasized that "(strictly) computational systems" depend on numerical data supplied by manufactured sensors and do not rely upon 'knowledge'." ([6], p. 88).

Then, Bezdek emphasized that "it is especially important and useful, in the context of the relationship between NNs and PR, to distinguish more carefully than usual what is meant by the term knowledge. Also the word 'artificial' raised trouble, when Bezdek wrote his paper: it seemed "much more properly applied in its usual context in AI than as it is currently used in NNs. Currently, it seems that the

[20] Note: BNN stands for *Biological* Neural Networks, BPR stands for *Biological* Pattern Recognition, and BI stands for *Biological* Intelligence, etc.

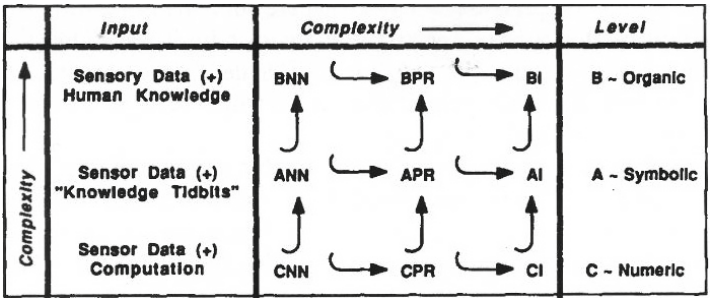


Fig. 6 Bezdek's "ABCs: Neural networks, pattern recognition, and intelligence", [6]

ANN is 'artificial' if it is not biological, that is, ANN is the complement of the BNN in the usual set-theoretic sense. However, I suggest a finer distinction between CNN and ANN, one that is connected to the term 'knowledge tidbits' in Figure 6." ([6], p. 88)

It was this sentence that was later used as a definition for CI in the introduction to the 1994 published book *Computational Intelligence: Imitating Life*, where the editors continued: "Artificial intelligence, on the other hand, uses what Bezdek calls 'knowledge tidbits'. Many NN's called 'artificial' should, Bezdek argues, be called computational NN's." ([59], p. v.)

In his contribution to the same book Bezdek responded to Fig. 6 more explicite: "The symbol ($\hookrightarrow$) in this figure means 'is a proper subset of'. For example, I am suggesting along the bottom row that CNNs $\subset$ CPR $\subset$ CI, and in the left column, that CNNs $\subset$ ANNs $\subset$ BNNs. As defined then, any computational system is artificial, but not conversely. So, I am definitiely suggesting that CI and AI are not synonyms. CI is in my view a proper subset of AI." In this paper Bezdek defined "CI systems" as follows:

> "A system is computationally intelligent when it: deals with only numerical (low-level) data, has pattern recognition components, does not use knowledge in the AI sense; and additionally when it (begins to) exhibit 1) computational adaptivity, 2) computational fault tolerance, 3) speed approaching humanlike turnaround and 4) error rates that approximate human performance." [7]

6 Conclusion

Concluding this chapter I would like to come back once again to Jim Bezdek's scheme in Fig. 6 that shows complexity levels in two dimensions: in Bezdek's words: "I think that A, B, and C correspond to three different levels of system complexity, which increase from left to right, and from bottom to top in this sketch." [7] When scientists try to create intelligent systems it means that this systems should perform (approximately) like *biological intelligent* (BI) systems. The way to reach this goal leads up and to the right by increasing complexity in both dimensions.

Bezdek pointed out that the concept "CI" is, however, only seductive as long as the concept of intelligence is no better defined than it currently is [7]. That means that in future times perhaps a similar scheme of complexity but showing more than two dimensions will be appropriate and paths to create an intelligent system — a "Thinking Machine" go labyrinthine ways to increase complexity. Science visions and science fictions will always try to show how such paths could appear but they will still base on their historical level of knowledge (tidbits)!

In December 1967, Isaac Asimov wrote a short text entitled "The Thinking Machine". The first sentence of this paper is: "The difference between a brain and a computer can be expressed in a single word: complexity." He argued that computers are programmed to solve problems and also human beings are programmed. Computers can only do what they are programmed to do; the same is true for humans, he wrote: "Our genes 'program' us the instant the fertilized ovum is formed, and our potentialities are limited by that 'program'." However, our program is that much more complex than computers have been in that time and still they are, but Asimov assumed that "if a computer can be made complex enough, [. . .] as complex as a human brain, it could be the equivalent of a human brain and do whatever a human brain can do." Moreover, his science vision — or is it science fiction? — says further that "we will perhaps build a computer that is at least complex enough to design another computer more complex than itself. This more complex computer could design one still more complex and so on and so on." [2] In this scenario it happens that computers "not only duplicate the human brain — but far surpass it." Then, there are two possibilies: "we ought to step aside" or the computers "push us aside".

Acknowledgements. Work leading to this paper was partially supported by the Foundation for the Advancement of Soft Computing Mieres, Asturias (Spain). I thank Karin Hutflötz for discussions and valuable comments, particularly to the story *Profession* and I thank Anett Hoppe and Anja Bachmann for support in writing this chapter.

References

[1] Asimov, I.: I, Robot. Gnome Press, New York (1950)
[2] Asimov, I.: The Thinking Machine. In: Farrell, E.J., Gage, T.E., Pfordresher, J., Rodrigues, R.J. (eds.) Science Fact/Fiction, pp. 90–91. Scott, Forseman & Company, Glenview (1974)
[3] Asimov, I., Warrick, P.S., Greenberg, M.H.: Machines That Think: The Best Science Fiction Stories About Robots and Computers. Holt, Rinehart, and Winston (1984)
[4] Bagley, J.D.: The behavior of adaptive systems which employ genetic and correlation algorithms. PhD thesis, University of Michigan (1967)
[5] Berkeley, E.C.: Giant Brains or Machines that think. John Wiley & Sons, Inc., New York (1949)
[6] Bezdek, J.C.: On the rrelationship between neural networks, pattern recognition and intelligence. International Journal of Approximate Reasoning 6, 85–107 (1993)
[7] Bezdek, J.C.: What is computational intelligence? In: [57], pp. 1–12

[8] Bonissone, P.P.: Soft computing: The convergence of emerging reasoning technologies. Soft Computing 1(1), 6–18 (1997)
[9] Cercone, N., McCalla, G.: Ten years of computational intelligence. Computational Intelligence 10(4), I (1994)
[10] Clute, J.: Science Fiction: The illustrated Encyclopedia, 1st American edn. Dorling Kindersley, New York (1995)
[11] Duch, W.: Quo vadis computational intelligence? In: Sincak, P., Vascak, J., Hirota, K. (eds.) Machine Intelligence: Quo Vadis?. Advances in Fuzzy Systems – Applications and Theory, vol. 21, World Scientific, Singapore (2004)
[12] Duch, W.: What is computational intelligence and where is it going? In: Duch, W., Mandziuk, J. (eds.) Challenges for Computational Intelligence. SCI, vol. 63, pp. 1–13. Springer, Heidelberg (2007)
[13] Fogel, L.G.: On the organization of intellect. PhD thesis, University of California, Los Angeles, CA, USA (1964)
[14] Haigh, T.: Technology's other storytellers: Science fiction as history of technology. In: Ferro, D.L., Swedin, E.G. (eds.) Science Fiction and Computing: Essays on Interlinked Domains, kindle Edition, NC, USA and London, UK, pp. 13–37. McFarland & Company, Jefferson (2011) cited: draft on Thomas Haigh's homepage, http://www.tomandmaria.com/tom/
[15] Haufe, R.: Design of a tit-tat-toe machine. Electrical Engineering 68, 885 (1949)
[16] Heims, S.J.: The Cybernetics Group. The MIT Press, Cambridge (1991)
[17] Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. The University of Michigan Press (1975)
[18] Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the USA 79(8), 2554–2558 (1982)
[19] Lohr, S.: Arthur W. Burks, 92, Dies; Early Computer Theorist. The New York Times (2008), http://www.nytimes.com/2008/05/19/technology/19burks.html
[20] Magdalena, L.: What is soft computing? revisiting possible answers. International Journal of Computational Intelligence Systems 3(2), 148–159 (2010)
[21] Maker, M.H.: Conference notes by Meg Houston Maker: McCarthy, J.: What was expected, what we did, and AI today [AI@50] filene auditorium (2006), http://www.megmaker.com/page/6/ for the conference see also www.dartmouth.edu/~ai50/homepage.html
[22] Martin, C.D.: The myth of the awesome thinking machine. Communications of the ACM 36(4), 120–133 (1993)
[23] McCarthy, J., Minsky, M.L., Rochester, N., Shannon, C.E.: A proposal for the Dartmouth summer research project on artificial intelligence (1955), http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html
[24] McCorduck, P.: Machines who Think. Freeman, San Francisco (1979)
[25] McCorduck, P.: Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence, 2nd edn. A. K. Peters, Natick (2004)
[26] McCulloch, W.S., Pitts, W.H.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5, 115–133 (1943)
[27] Minsky, M., Papert, S.: Perceptrons. The MIT Press, Cambridge (1969)
[28] Mitchell, M.: Complexity: A Guided Tour. Oxford University Press, New York (2009)

[29] Moor, J.: The Dartmouth college artificial intelligence conference: The next fifty years. AI Magazine 27(4), 87–91 (2006)
[30] von Neumann, J.: The Computer and the Brain. Yale University Press, New Haven (1958)
[31] von Neumann, J.: First draft of a report on the edvac. IEEE Annals of the History of Computing 15(4), 27–75 (1993)
[32] Newell, A., Simon, H.A.: The logic theory machine. IRE Transactions on Information Theory IT-2(3), 61–79 (1956)
[33] Poole, D., Mackworth, A., Goebel, R.: Computational Intelligence: A Logical Approach. Oxford University Press, New York (1998)
[34] Rechenberg, I.: Evolutionsstrategie. Friedrich Frommann, Stuttgart / Bad Cannstatt, Germany (1973)
[35] Robinson, K.S.: Notes for an essay on cecelia holland. Foundation 40, 54–61 (1987)
[36] Rojas, R., Hashagen, U. (eds.): The First Computers: History and Architectures. The MIT Press, Cambridge (2000)
[37] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review 65(6), 386–408 (1958)
[38] Rumelhart, D.E., McClelland, J.L., the PDP research group: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, 2 volumes. The MIT Press, Cambridge (1986)
[39] McCulloch, W.S.: Why the mind is in the head. In: Jeffress, L.A. (ed.) Cerebral Mechanisms in Behavior: The Hixon Symposium, pp. 42–57. John Wiley & Sons, Inc., New York (1951)
[40] Schwefel, H.P.: Evolutionsstrategie und numerische optimierung. Dissertation, Technische Universität Berlin (1975)
[41] Seising, R.: Interview with L. A. Zadeh on July, 26, see [42] (2000) (unpublished)
[42] Seising, R.: Interview with L. A. Zadeh on June 15, see [42] (2001) (unpublished)
[43] Seising, R.: The Fuzzification of Systems: The Genesis of Fuzzy Set Theory and Its Initial Applications – Developments up to the 1970s. Springer, New York (2007)
[44] Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal 27, 379–423 and 623–656 (1948)
[45] Shannon, C.E., McCarthy, J. (eds.): Automata studies. Princeton University Press, Princeton (1956)
[46] Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. In: Proceedings of the London Mathematical Society, 2, vol 42, pp. 230–265 (1936-1937); with corrections from Proceedings of the London Mathematical Society, series 2, vol. 43, pp. 544–546 (1937)
[47] Turing, A.M.: Computing machinery and intelligence. Mind 49(236), 433–460 (1950)
[48] Wiener, N.: Cybernetics or Control and Communication in the Animal and the Machine. Hermann & Cie., The Technology Press, and John Wiley & Sons, Inc., Cambridge, MA and New York (1948)
[49] Zadeh, L.A.: Thinking machines – a new field in electrical engineering. Columbia Engineering Quarterly, 12–13, 30–31 (1950)
[50] Zadeh, L.A.: Fuzzy sets. Information and Control 8, 338–353 (1965)
[51] Zadeh, L.A.: Fuzzy languages and their relation to human and machine intelligence. In: Proc. of Intl. Conf. on Man and Computer, Bordeaux, France, pp. 130–165 (1972)
[52] Zadeh, L.A.: Making computers think like people. IEEE Spectrum 8, 26–32 (1984)
[53] Zadeh, L.A.: Foreword. Applied Soft Computing 1(1), 1–2 (2001)

[54] Zadeh, L.A.: A new direction in ai: Toward a computational theory of perceptions. AI-Magazine 22(1), 73–84 (2001b)
[55] Zadeh, L.A.: My life and work – a retrospective view. Applied and Computational Mathematics 10(1), 4–9 (2011)
[56] Zadeh, L.A., Desoer, C.A.: Linear System Theory: The State Space Approach. McGraw-Hill Book Company, New York (1963)
[57] Zimmermann, H.J.: Editorial. Fuzzy Sets and Systems 69(1), 1–2 (1995)
[58] Zurada, J.M., Marks, R.J., Robinson, C.J. (eds.): Computational Intelligence Imitating Life. IEEE Press, Los Alamitos (1994)
[59] Zurada, J.M., Marks, R.J., Robinson, C.J.: Introduction. In: [57], pp v–xi (1994b)

Part III
Uncertainty in Knowledge-Based Systems

Markov Network Revision: On the Handling of Inconsistencies

Jörg Gebhardt, Aljoscha Klose, and Jan Wendler

Abstract. Graphical models are of high relevance for complex industrial applications. The Markov network approach is one of their most prominent representatives and an important tool to structure uncertain knowledge about high-dimensional domains in order to make reasoning in such domains feasible. Compared to *conditioning* the represented probability distribution on given evidence, the important belief change operation called *revision* has been almost entirely disregarded in the past, although it is of utmost relevance for real world applications. In this paper we focus on the problem of *inconsistencies* during revision in Markov networks. We formally introduce the revision operation and propose methods to specify, identify, and resolve inconsistencies. The revision and its inconsistency management has proven to be successful in a complex application for item planning and capacity management in the automotive industry at Volkswagen Group.

1 Introduction

Today's scientific and economic problems are often characterised by a large number of variables. With a sufficiently high number of variables, the complexity of these problems grows quickly, so that analyses and reasoning processes become increasingly difficult. For this reason, lossless or approximating decomposition techniques are often necessary in order to efficiently cope with high dimensionalities. Decomposition is achieved by making use of (conditional) independencies between variables. Graphical models [15, 12, 1] have established themselves as one of the most popular tools to structure uncertain knowledge in this way, so that inference becomes feasible [14, 9]. Their most prominent representatives are Bayesian networks [14], which are based on directed graphs and conditional probability distributions, as well as Markov networks [13], which refer to undirected graphs and marginal probability distributions or factor potentials.

Jörg Gebhardt · Aljoscha Klose · Jan Wendler
ISC Gebhardt, Celle, Germany
e-mail: info@isc-gebhardt.de

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 153–165.
springerlink.com

When dealing with graphical models several non-trivial operations have to be considered. For the first step of knowledge representation, one needs learning [4, 11] and data fusion algorithms to get an appropriate structure and the initial distributions of a network. Further knowledge processing on a given network is realized, for example, by information retrieval, belief change, and inference operations, respectively [3, 4, 5].

The most discussed knowledge processing operation in the field of probabilistic graphical models is **focusing**, which can be achieved by performing any kind of evidence-driven conditioning on a set of input variables and propagating the new information. Instantiation of variables as it is usually implemented in diagnostic tools, can be considered as a special case of this operation, with all the probability mass assigned to one value per given variable.

It is surprising that other essential operations well-known from uncertainty management in knowledge-based systems seem to be overlooked in the scientific community of graphical modeling: They concern the two almost complementary operations of revision and updating, respectively. Compared to focusing, these operations are not restricted to pure information retrieval and simulation aspects, but reflect the task of belief change.

Revision refers to an alteration of the represented probability distribution within the frame of an existing model structure, i.e. although the probability of a state (element of the common domain) may be changed in the revision process, it is required that forbidden states (having a zero probability) do not change. Revision is performed by locally introducing new distributions into the Markov network. Like with focusing, local modifications of distributions are propagated. But in contrast to the operations used in information retrieval, changes made during revision are permanent, as the modified distributions replace those already stored in the model. The alterations to the model are the least ones required to integrate the new probability assignments. Therefore the maximum of the probabilistic interaction structure already represented in the model is preserved, which coincides with the so-called principle of minimal change [5].

If multiple local distributions in the network have to be modified, the desired revision is achieved by propagating the new assignments one after another (iterative proportional fitting [15]). Since any local change may affect other areas in the model, processing one of the assignments may invalidate part of the models' adaptations to previous assignments in the sequence. However, by iterating the process the model will often converge to a state of stable compromise, consistent with all assignments.

Nevertheless, if the assignments are in conflict with each other or affect zero probabilities of the initial distribution, the whole revision process will remain unstable and equilibrium cannot be achieved. In the first case there can be no consistent, accurate and complete model for contradictory evidence. In the latter case a solution can be achieved by applying an updating operator.

Updating is complementary to a revision operation in the sense that this operation locally introduces a new probabilistic interaction structure to a model, which means that it changes probabilities from zero to positive values and therefore defines

new probabilistic dependencies between the involved variables. For this reason, it does, of course, not follow the principle of minimal change.

In the following contribution we focus our interest on the topic of handling inconsistencies in revision problems. The underlying research was triggered by an application at the automobile manufacturer Volkswagen Group, where ISC Gebhardt established Markov networks for the development of a world-wide software system for item planning and capacity management [2, 7, 6].

The paper is structured as follows: Section 2 introduces the complex item planning problem. In Section 3 we establish definitions to specify and define the revision problem. We discuss revision and identify inconsistency problems which may arise. In Section 4 we focus on inconsistencies, thereby differentiating between inner and outer consistency. Furthermore we introduce the degree of inconsistency of a revision problem. Finally, in Section 5 we present practical solutions to handle inconsistencies in a complex domain.

2 Real-World Application

2.1 Item Planning at Volkswagen Group

In contrast to many competing car manufacturers, Volkswagen Group favours a marketing policy that provides a maximum degree of freedom in choosing individual specifications of vehicles. That is, considering personal preferences, a customer may select from a large variety of options, each of which is taken from a so-called item family that characterises a certain line of equipment. Body variants, engines, gearshifts, door layouts, seat coverings, radios and navigation systems reflect only a small subset of the whole range of item families. In case of the VW Golf – Volkswagen's most popular car class – there are about 200 families with typically 4 to 8 values each, and a total range of cardinalities from 2 up to 150.

Of course, not all of the possible instantiations of item variables lead to valid vehicle configurations, since technical rules, restrictions in manufacturing and sales requirements induce a common rule system that limits the item combinations. Nevertheless, dealing with more than 10,000 technical rules in the Golf class and even more rules delivered by the sales programs for the special needs of different countries, there remains a huge number of correct vehicle specifications. In fact, compared to a total of 910,000 Golf cars in 2011, one can find only a small number of vehicles within the whole production line that have identical specifications.

The major aim of the productive system EPL (EigenschaftsPLanung, German for item planning) at Volkswagen Group was the development and implementation of a software solution that supports item planning, parts demand calculation, and capacity management with the aim of short-term as well as medium-term forecasts up to 24 months of future vehicle production.

In order to achieve high quality of planning results, all relevant information sources have to be considered, namely rules for the correct combination of items into complete vehicle specifications, samples of produced vehicles as a reflection of

customers' preferences, market forecasts that lead to revision assignments of modified item rates for planning intervals, capacity restrictions, and production programs that fix the number of planned vehicles.

With respect to the logistics view, the most essential result to assess of the item planning process are the rates of all those item combinations that are known to be relevant for the demand calculation of parts, always related to a certain vehicle class in a certain planning interval. The importance of these item combinations arises from the fact that a vehicle can be interpreted as a large set of installation points, each of which is characterised by a set of alternative parts for the corresponding location. Which of the alternative parts has to be chosen at an installation point depends on its installation condition that can be specified by an item combination. Of course, at each installation point, all occurring installation conditions have to be disjoint, and their disjunction has to form a tautology. That is, given any correct vehicle specification, for each installation point we obtain a unique decision which of its alternative parts has to be used.

In the context of the Golf class, we find a total of about 70,000 different item combinations required as installation conditions for the whole set of installation points. The data structure that lists all installation points, their installation conditions, and the quantities of the referenced parts, is called a variants-related bill of material. The task of predicting the total demand of a certain part with respect to a future planning interval is to sum up the demands over all of its installation points. The demand at any installation point results from multiplying the rate of the item combination that represents its installation condition with the quantity and the total number of vehicles intended to be produced in the respective planning interval.

We conclude that calculating parts demand is reduced to a simple operation, whenever the rates of all involved item combinations can be computed.

2.2 Markov Network Model and Revision Operator

The first step in the project EPL was to search for an appropriate planning model that supports a decomposed representation of the qualitative and quantitative dependency structure between item families. We had to take into account that we deal with a finite set of discrete item variables, as well as that we get conditional independences induced by the given rule systems and customers' preferences.

Since logical rule systems can be transformed into a relational setting, and rates for item combinations may be identified as (frequentistic or subjective) occurrence probabilities within a large sample of historical or predictably valid vehicle specifications, Markov networks turned out to be the most promising environment to satisfy the given modelling purposes.

Once a basic prior Markov network for a certain planning interval has been generated, it becomes subject to a variety of planning operations which involve marketing and sales stipulations (e.g., installation rate of comfort navigation system increases from 20 % to 35 %) and capacity restrictions from logistics (e.g., maximum availability of seat coverings in leather is 5,000). These quantitative input data

are strongly related to the planning interval itself and therefore not learnable, neither from historical data nor from the non-probabilistic rule system. They typically consist of predicted installation rates or absolute demands for single items, sets of items, or (sets of) item combinations, and are frequently related to refined planning contexts (e.g., VW Golf with all-wheel drive for the US market).

In mathematical terms, this sort of additional information leads to competing partial or total changes of selected (conditional) low-dimensional probability distributions in the Markov network. Such changes can be interpreted as the basis for a revision operation, where a prior state of knowledge (represented by the initial Markov network) given new information (which is the new set of probability distributions) is revised to a posterior state of knowledge. The new information is thereby incorporated in the sense of the *principle of minimal change* [5].

In terms of the probabilistic framework, the task is to calculate a posterior Markov network that satisfies the new distribution conditions, only accepting a minimal change of the quantitative interaction structures of the underlying prior distribution.

3 Revision in Markov Networks

Before starting the discussion about inconsistencies in Markov networks we need to specify the revision problem and define its solution. Furthermore we reflect how the proposed solution can be calculated and discuss which problems will arise during the revision operation due to the complexity and human factor in real world applications.

3.1 Definitions

Suppose that we are given a **Markov network** $M = (H, \Psi)$ which represents a joint probability distribution $P(V)$ on a set $V = \{X_1, ..., X_n\}$ of variables with finite domains $\Omega(X_i)$, $i = 1, ..., n$. We assume that $H = (V, \{C_1, ..., C_m\})$ denotes a **hypertree** of which the C_i are the (maximal) cliques and $\Psi = (P(C_j))_{j=1}^{m}$ a **family of probability distributions** defined on the (maximal) cliques of H. In this setting, H and its associated **undirected dependency graph** $G(H)$ reflect the conditional independencies between the involved variables, and Ψ shows the resulting **factorization property** $P(V) = \prod_{j=1}^{m} P(C_j)/P(S_j)$, where S_j symbolize the **separators** in some representation of H as a **tree of cliques**.

In addition, let $\Sigma = (\sigma_s)_{s=1}^{S}$ be a so-called **revision structure** that consists of **revision assignments** σ_s, each of which is referred to a (conditional) assignment scheme $(R_s|K_s)$ with a **context scheme** K_s, $K_s \subseteq V$, and a **revision scheme** R_s, where $\varnothing \neq R_s \subseteq V$ and $K_s \cap R_s = \varnothing$. We assume that σ_s is specified by a set of assignment components $P^*(\rho_s^{(k,l)}|\kappa_s^{(k)})$, where $\kappa_s{}^{(k)}$ is called its **context component** and $\rho_s{}^{(k,l)}$ its **revision component**, respectively. The context components are expected to specify a partitioning of $\Omega(K_s)$, and for each $k \in \{1, ..., k^*(s)\}$, the set

$\{\rho_s^{(k,l)} | l = 1,...,l^*(s,k)\}$ forms a partitioning of $\Omega(R_s)$. Hence, each revision assignment specifies a modified probability distribution referred to the scheme $(R_s|K_s)$, separable into independent modifications of the distributions $P(R_s|\kappa_s^{(k)})$, given by the assignment components $P^*(\rho_s^{(k,l)}|\kappa_s^{(k)})$, $l = 1,...,l^*(s,k)$. In case of the empty scheme $K_s = \varnothing$, we deal with an assignment of the (non-conditioned) probabilities $P^*(\rho_s^{(l)})$.

Finally, we suppose that for all $s = 1,...,S$ there are cliques $C(s) \in \{C_1,...,C_m\}$ such that $K_s \cup R_s \subseteq C(s)$. This guarantees that we do not have cross-over dependencies between cliques, which may not be expressible in the structure of the given Markov network.

Definition 1 (Solution of revision problems). Let $M = (H,\Psi)$ be a Markov network with associated joint probability distribution $P(V)$. Furthermore, let $\Sigma = (\sigma_s)_{s=1}^S$ be a revision structure.

A probability distribution $P_\Sigma(V)$ is called **solution of the revision problem** $(P(V),\Sigma)$, if and only if the following conditions hold:

(R1) **Revision assignments are satisfied:**

$$(\forall s \in \{1,...,S\})(\forall k \in \{1,...,k^*(s)\})(\forall l \in \{1,...,l^*(s,k)\}) \left(P_\Sigma(\rho_s^{(k,l)}|\kappa_s^{(k)}) = P^*(\rho_s^{(k,l)}|\kappa_s^{(k)})\right)$$

(R2) **Preservation of interaction structure:**
Except from the modifications induced by the revision assignments, $P_\Sigma(V)$ preserves all probabilistic dependencies of $P(V)$.

3.2 Discussion

Essentially, the required preservation of the interaction structure coincides with the decision-theoretical presupposition that the revision operator does not modify the cross product ratios of conditional events outside the influence areas of the revision assignments (**principle of minimal change**).

It can be proven (see, f.e. [8]) that in case of existence, the solution of the revision problem $(P(V),\Sigma)$ is uniquely defined. $P_\Sigma(V)$ can be calculated as the limit probability distribution if the revision procedure of **iterative proportional fitting** with parameters Σ is applied to the initial distribution $P(V)$.

From a practical point of view, in most cases of real world applications of sufficient complexity, we have to take into account that revision problems $(P(V),\Sigma)$ specified by human experts are not solvable. The reason for this observation is the fact that revision structures $\Sigma = (\sigma_s)_{s=1}^S$ tend to contradict some of the restrictions given by the zero values of the initial probability distribution $P(V)$. Note that assignment components $P^*(\rho_s^{(k,l)}|\kappa_s^{(k)}) > 0$ may induce to change some probabilities $P(\omega) = 0$ to a strictly positive value. This kind of modification is not conform to the dependency preservation requirement of the revision operator, as

zero probabilities show the absence of any interaction structure. Hence, a resulting probability $P_\Sigma(\omega) > 0$ would introduce a new interaction structure, which is the typical focus of the (in some sense complementary) updating operations.

Resulting probabilities $P_\Sigma(\omega) > 0$ may be introduced directly by only one revision assignment σ_s (which can be easily detected and coped with) or by any subset of Σ. In the latter case the revision structure Σ contains inconsistencies which cannot be detected and dealt with by trivial means. In order to deal with such inconsistencies we need first to analyse their properties and categorise these inconsistencies.

4 Categorisation of Inconsistencies

As already mentioned in the previous section, inconsistencies may occur during revision. Inconsistencies can be roughly classified into two categories. In this section we will differentiate between inner and outer (in-)consistency. Inner consistency is a property of a revision structure alone whereas outer consistency always depends on the initial distribution $P(V)$, especially its zero values. Moreover, we introduce inner and outer inconsistency criteria for revision problems which finally allows us to determine the degree of inconsistencies.

4.1 Definitions

In order to handle the typical inconsistencies which arise during the revision, we introduce the following definitions:

Definition 2 (Inner consistency). Let $(P(V), \Sigma)$ be a revision problem. A revision structure Σ shows the property of **inner consistency**, if and only if there exists a probability distribution that satisfies the revision assignments of Σ.

Note that this definition is conform to the condition (R1) of definition 1.

Definition 3 (Outer inconsistency). Let $(P(V), \Sigma)$ be a revision problem with the property of inner consistency. $(P(V), \Sigma)$ shows the property of **outer inconsistency**, if and only if there is no solution of this revision problem.

Definition 4 (ε-modification of a revision problem). Let $(P(V), \Sigma)$ be a revision problem, and let ε be a (sufficiently small) positive real number. Furthermore, assigning $r := |\{\omega \in \Omega(V) | P(\omega) = 0\}|$, let

$$P_\varepsilon(\omega) \stackrel{Df}{=} \begin{cases} P(\omega) \cdot (1 - r\varepsilon), & \text{if } P(\omega) > 0 \\ \varepsilon, & \text{if } P(\omega) = 0 \end{cases}$$

Then, $(P_\varepsilon(V), \Sigma)$, is called the **$\varepsilon$-modification** of $(P(V), \Sigma)$.

We now present some results that are useful for the recognition and handling of inconsistencies of revision problems.

4.2 Inner Inconsistencies

Theorem 1 (Inner inconsistency criterion for revision problems). *A revision problem* $(P(V),\Sigma)$ *shows the property of inner consistency, if and only if the revision procedure (iterative proportional fitting) applied to its* ε*-modification* $(P_\varepsilon(V),\Sigma)$ *converges to a limit distribution* $P_\varepsilon^\Sigma(V)$.

Corollary 1 (Sufficient condition for revisability). *In case of a strictly positive distribution* $P(V)>0$ *and inner consistency of its structrue* Σ *of assignments, there always exists the uniquely determined solution* $P^\Sigma(V)\equiv P_\varepsilon^\Sigma(V)$ *of the revision problem* $(P(V),\Sigma)$.

To achieve inner consistency, one may restrict the revision structure Σ, so that it fulfills the following conditions:

1. $(\forall s\in\{1,...,S\})(\forall t\in\{1,...,S\})(s\neq t\Rightarrow R_s\cap R_t=\varnothing)$
2. $(\exists(V,<))(\forall s\in\{1,...,S\})(\rho\in R_s\Rightarrow(\forall\kappa\in K_s)(\kappa<\rho))$

The first condition forbids that two different assignment components (with possibly different context schemes) specify parts of the same revision scheme. The second condition ensures that no cyclic dependencies between the assignment components are possible.

However, these conditions for the revision structure are quite restricting: Considering the presupposed distinction of context and revision schemes ($K_s\cap R_s=\varnothing$) as well as the inclusion of $K_s\cup R_s$ in one of the cliques, it turns out that (1) and (2) lead to a so-called chain graph which reflects the dependencies induced by the given revision structure. This means that all dependencies of the involved variables may be specified with the aid of a composition of directed acyclic graphs and undirected graphs. For practical purposes, whenever possible, it is desirable to establish such a dependency graph. An alternative is to use techniques of locating and removing (inner) inconsistencies rather than preventing them (see section 5).

4.3 Outer Inconsistencies

Lemma 1 (Theoretical outer inconsistency criterion for revision problems). *Given the inner consistency of its structure* Σ *of assignments, we observe an outer inconsistency of a revision problem* $(P(V),\Sigma)$*, if and only if*

$$(\exists\omega\in\Omega(V))(P(\omega)=0\ \wedge\ P^*(\omega)>0)$$

holds for all probability distributions $P^*(V)$ *that satisfy the revision assignments (R1).*

Theorem 2 (Practical outer inconsistency criterion for revision problems). *Given the inner consistency of its structure* Σ *of assignments, we observe an outer inconsistency of a revision problem* $(P(V),\Sigma)$*, if and only if*

$$(\exists \omega \in \Omega(V))(P(\omega) = 0 \Rightarrow P_{\varepsilon}^{\Sigma}(\omega) >> P_{\varepsilon}(\omega))$$

is satisfied.

As a consequence, given inner consistency, the outer inconsistency proof for a particular revision problem $(P(V), \Sigma)$ can be obtained by application of the revision procedure to the ε-modification $(P_{\varepsilon}(V), \Sigma)$ of this revision problem, and testing the limit distribution $P_{\varepsilon}^{\Sigma}(V)$ with respect to the outer inconsistency criterion.

4.4 Degree of Inconsistencies

Further investigations on mass flows in inconsistency situations finally lead to the following theorem that gives a basis to handle inconsistencies of non-solvable revision problems:

Theorem 3. *Given the assumptions of the previous theorem, let*

$$Inconsistent\_tuples(P(V), \Sigma) \stackrel{Df}{=} \left\{\omega \in \Omega(V) | P(\omega) = 0 \wedge P_{\varepsilon}^{\Sigma}(\omega) >> P_{\varepsilon}(\omega)\right\}$$

denote the set of all tuples that are involved in the outer inconsistencies of a revision problem $(P(V), \Sigma)$. Then, this set consists of all invalid tuples ω that need significant probability mass flows (quantified by $P_{\varepsilon}^{\Sigma}(\omega)$) from ω to any valid tuples in order to remove the existing inconsistencies.

After inconsistent tuples are located one can determine the degree of the inconsistency which is given by

$$Inconsistency\_mass(P(V), \Sigma) \stackrel{Df}{=} \sum \left\{P_{\varepsilon}^{\Sigma}(\omega) | \omega \in Inconsistent\_tuples(P(V), \Sigma)\right\}.$$

$Inconsistency\_mass(P(V), \Sigma)$ reflects the whole probability (inconsistency) mass which has to be transferred to any tuples of $\Omega(V) - Inconsistent\_tuples(P(V), \Sigma)$.

5 Practical Solutions to Handle Inconsistencies

With the analysis of inconsistencies from the last section one can identify the inconsistent tuples and distribute the inconsistency mass to other tuples using the ε-modification of a revision problem. However, applying the ε-modification makes it necessary to hold in memory all tuples within the cliques C_j, even the tuples which had a zero probability before. Note that in the original revision problem zero probabilities do not need to be represented explicitly.

In our application domain the sizes of average cliques and largest cliques differ among the automobil models. In typical automobile models (like the Golf class) the largest cliques contain about 40,000 non-zero tuples, but the maximal theoretical size[1] of these cliques is greater than 10^{14}, which makes it infeasible to apply the

[1] Size of all tuples including the zero-value tuples.

ε-modification to these cliques. Therefore practical solutions are needed to handle inconsistencies.

The main idea for the practical solution presented in this paper is to prioritise and group the revision assignments as well as apply several revisions[2] with consistency checking (and adaptation of revision assignments if necessary) until all revision assignment groups are incorporated. With this strategy it is possible to locate and remove all inconsistencies without the need to differentiate between inner and outer inconsistencies anymore.

In the following we will speak about inconsistencies between revision assignments. Please note that the intial distribution $P(V)$, especially its zero-values, is always part of such inconsistencies but will be regarded as not adaptable to solve inconsistencies and therefore $P(V)$ is not mentioned all the time.

5.1 Prioritising and Grouping the Revision Assignments

Given a potentially inconsistent revision problem $(P(V), \Sigma)$ one can often specify which revision assignments σ_s are more important than others, so that in case of an inconsistency between these revision assignments only the least important one should be adapted.

However, sometimes it is impossible to decide which one of two revision assignments is more important. In fact it might be needed that two or more revision assignments get the same priority. Such revision assignments are grouped together. In case of an inconsistency within such a group all its revision assignments should be adapted according to the principle of minimal change.

The set of revision assignments is divided into n partitions $\mathbb{S}_i$, so that $\Sigma = \bigcup_{i=1}^{n} \mathbb{S}_i$ and $\mathbb{S}_i \neq \varnothing$, $\mathbb{S}_i \cap \mathbb{S}_j = \varnothing$ for any $1 \leq i, j \leq n$ with $i \neq j$.

5.2 Iterative Revision with Consistency Checking

After the revision assignments are grouped and ordered we can start with an empty set Σ_0 which is consistent with the initial probability distribution $P(V)$. In each iteration we take the consistent set Σ_{i-1} $(1 \leq i < n)$ and perform a meta revision by adding the revision assignments of $\mathbb{S}_i$. This meta revision results in a new consistent set Σ_i where the revision assignments of $\mathbb{S}_i$ are adapted (where necessary) to achieve consistency with Σ_{i-1}.

Each meta revision operates using (up to) two phases. In the first phase a single revision is performed on the set $\Sigma_{i-1} \cup \mathbb{S}_i$. If this revision converges, the set is consistent, otherwise the revision assignments of $\mathbb{S}_i$ introduce an inconsistency. This inconsistency is resolved by applying partition mirrors to $\mathbb{S}_i$. By applying these partition mirrors, the revision converges and the revision assignments $\sigma_s \in \mathbb{S}_i$ are adapted to σ_s^* if necessary. The main idea of partition mirrors is to mirror variables into the network structure, couple the states of these variables to their origins by

[2] For detailed information about a single revision step, please see [8].

suitable initial distributions, and reformulate the assignments in order to set probabilities to these new variables. For detailed information about partition mirrors see [10].

After n steps every revision assignment has been tested for consistency with the other revision assignments. In case of inconsistencies the least important revision assignments have been adapted so that we finally have a consistent set Σ_n of revision assignments.

The resulting algorithm reads:

$i := 0$; $\Sigma_i := \varnothing$; $(*$ initializing empty set $*)$
repeat $(*$ iterative meta revision $*)$
$i := i + 1$ $(*$ iteration counter increment $*)$
phase 1: test consistency
  do single revision with $\Sigma_{i-1} \cup \mathbb{S}_i$
  if probability distribution converges:
    $\Sigma_i := \Sigma_{i-1} \cup \mathbb{S}_i$
  else {
phase 2: applying partition mirrors
  do single revision with $\Sigma_{i-1} \cup \mathbb{S}_i$ applying partition mirrors for $\mathbb{S}_i$
  $\Sigma_i := \Sigma_{i-1} \cup \mathbb{S}_i^*$ where $\mathbb{S}_i^*$ contains adapted revision assignments }
until $i = n$ $(*$ all revision assignments incorporated $*)$

The result of each iteration (meta revision) is a consistent set of revision assignments Σ_i as well as a modified probability distribution $P^{\Sigma_i}(V)$. The i-th meta revision can be performed on either the initial probability distribution $P(V)$ or on the resulting distribution of the prior iteration $P^{\Sigma_{i-1}}(V)$. Using the resulting distribution of the iteration before, one benefits from the already incorporated assignments Σ_{i-1}.

With highly inconsistent revision assignments it may be practical to skip phase one of the meta revision and assume that the assignments in $\mathbb{S}_i$ introduce a new inconsistency. By skipping phase one, calculation time can be saved but the application of partition mirrors also introduces additional calculation time.

If inconsistencies are rare it is possible to further group the revision assignments so that the number of single revisions can be reduced. In case of an inconsistency with a set $\bigcup_{\text{any}\,i} \mathbb{S}_i$, this set has to be divided again to locate the $\mathbb{S}_i$ which introduces the inconsistency.

6 Conclusion

In this paper we analysed inconsistencies which may occur during Markov network revision. We identified two inconsistency categories, namely inner inconsistency and outer inconsistency. With the help of the *ε-modification of a revision problem* we analysed the differences and special properties of these two inconsistency categories. Consequently, it was possible to specify the degree of inconsistencies based on the number of tuples involved as well as their probability mass. Not only theoretical considerations to identify and resolve inconsistencies are presented, although

a practical approach to handle inconsistencies was proposed in this work. This approach is based on prioritising and grouping revision assignments so that iterative revision operations can be used to identify and resolve inconsistencies efficiently.

Automatically resolved inconsistencies are very beneficial for the user since it reduces the manual effort drastically. However, sometimes additional information is needed in order to explain to the user why some assignments have been adapted. Therefore an automatically generated explanation of inconsistencies would be very helpful. Such an explanation could be determined in two steps. In the first step a minimal set of revision assignments causing the inconistency could be generated. In the second step an argumentation line could be given in order to explain the inconsistency. The automatic explanation of inconsistencies seems to be more complicated than expected (especially the second step). This task is subject to further research.

Acknowledgements. The authors thank Dr. Heinz Detmer and Dirk Schäfer from Volkswagen Group for supporting our work in the context of the software system EPL. Furthermore we give thanks to Katharina Tschumitschew, Fabian Schmidt, and Martin Försterling for their useful advice when writing this contribution.

References

[1] Borgelt, C., Kruse, R.: Graphical Models—Methods for Data Analysis and Mining. J. Wiley & Sons, Chichester (2002)
[2] Detmer, H., Gebhardt, J.: Markov-Netze für die Eigenschaftsplanung und Bedarfsvorschau in der Automobilindustrie. KI – Künstliche Intelligenz (03/01) (2001) (in German)
[3] Gabbay, D., Smets, P. (eds.): Handbook of Defeasable Reasoning and Uncertainty Management Systems. Belief Change, vol. 3. Kluwer Academic Press, Netherlands (1998)
[4] Gabbay, D., Smets, P. (eds.): Handbook of Defeasable Reasoning and Uncertainty Management Systems. Abductive Reasoning and Learning, vol. 5. Kluwer Academic Press, Netherlands (2000)
[5] Gärdenfors, P.: Knowledge in Flux: Modeling the Dynamics of Epistemic States. MIT Press, Cambridge (1988)
[6] Gebhardt, J., Kruse, R.: Knowledge-Based Operations for Graphical Models in Planning. In: Godo, L. (ed.) ECSQARU 2005. LNCS (LNAI), vol. 3571, pp. 3–14. Springer, Heidelberg (2005)
[7] Gebhardt, J., Detmer, H., Madsen, A.: Predicting Parts Demand in the Automotive Industry – An Application of Probabilistic Graphical Models. In: Proc. Int. Joint Conf. on Uncertainty in Artificial Intelligence UAI 2003, Bayesian Modelling Applications Workshop, Acapulco, Mexico (2003)
[8] Gebhardt, J., Borgelt, C., Kruse, R., Detmer, H.: Knowledge revision in Markov Networks. Special Issue "From Modelling to Knowledge Extraction" 11(2-3), 93–107 (2004)
[9] Jensen, F.: An Introduction to Bayesian Networks. UCL Press, London (1996)
[10] Klose, A., Wendler, J., Gebhardt, J., Detmer, H.: Resolution of inconsistent revision problems in Markov Networks. In: 6th International Conference on Soft Methods in Probability and Statistics. AISC. Springer (2012)

[11] Kollar, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
[12] Lauritzen, S.L.: Graphical Models. Oxford University Press (1996)
[13] Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society, Series B 2(50), 157–224 (1988)
[14] Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, 2nd edn. Morgan Kaufman, New York (1992)
[15] Whittaker, J.: Graphical models in applied multivariate statistics. Wiley, Chichester (1990)

Feedback-Driven Design of Normalization Techniques for Biological Images Using Fuzzy Formulation of a Priori Knowledge

Arif ul Maula Khan, Markus Reischl, Brigitte Schweitzer,
Carsten Weiss, and Ralf Mikut

Abstract. In digital imaging, a normalization procedure is an important step for an efficient and meaningful analysis of any random image dataset. The original intensity information in a digital image is mostly distorted due to imperfect acquisition conditions resulting in the shading phenomenon. Additionally, the high contrast of gray values present in an image also imparts a bias to retrieved gray values. Consequently, image processing goals such as segmentation and cell classification are adversely affected by aforementioned factors. In many microscopic imaging applications, *retrospective* shading correction methods are more commonly used as opposed to *prospective* methods in order to remove unwanted shading effects. The objectives of a normalization process, for one, can be rescaling of pixel values to a desired range while disregarding outliers and noisy background pixels. To counter shading effects, robust normalization techniques based on the adaptation of normalization parameters should be devised. We propose a feedback-based automatic image normalization technique that incorporates the evaluation criterion for its effectiveness based on image processing goals such as segmentation. Such a technique employs surface fitting of the available image pixel values to structures of a given family of function (such as polynomials) describing the spatial intensity variation of that image. It incorporates fuzzy formulation of criteria for normalization evaluation as an internal consistency check, while including post-segmentation results based on *a priori* segmentation knowledge at the same time. Results from a biological dataset consisting of images showing normal and dying cells are included to elucidate the effectiveness of the proposed scheme by automatically adapting normalization parameters.

1 Motivation and Overview

Digital images are subject to a diversity of unwanted distortions that are inevitably linked to the acquisition conditions, pre-processing and storage. In the realm of

Arif ul Maula Khan · Markus Reischl · Brigitte Schweitzer · Carsten Weiss · Ralf Mikut
Karlsruhe Institute of Technology, 76344 Eggenstein-Leopoldshafen, Germany
e-mail: ralf.mikut@kit.edu

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 167–178.
springerlink.com

image processing, normalization alludes to the retrieval of original inherent information (i.e. intensity values) in an image. This is achieved by manipulation of pixel values (i.e. corresponding image intensity values) in order to obtain a desired range of pixel values satisfying certain desired image processing goals such as segmentation.

The optimal parameters of an image segmentation procedure are often affected by side effects (e.g. blurriness, noise, inconsistent background illumination etc.) [9]. These side effects cause distortions in image intensities, which in turn cause loss of targeted information present in an image. For instance, microscopic images are often corrupted by erroneous intensity variation on account of inherent shortcomings of the image formation process. This phenomenon of intensity variation in literature is described using terms such as shading, intensity inhomogeneity, intensity non uniformity, bias field and gain field [5, 11]. While manual image analysis might be less prone to invalid interpretation of images in presence of such spurious intensity variation effects, automatic image analysis is likely to get riddled with such effects [10]. Consequently, in the absence of an appropriate image normalization procedure, eventual image processing goals such as image segmentation and object classification would be affected unfavorably. The aim is to introduce sophisticated image normalization techniques such that desired image processing goals could be achieved efficiently. Moreover, an adequate formulation of criteria to evaluate the performance of a normalization technique should be done in order to quantify the normalization outcome.

Currently, there are myriad of shading correction methods on the horizon. Shading can be roughly categorized as either object-independent or object-dependent [5]. The former is originated from certain shortcomings in the image acquisition process and is independent of the imaged object while the latter is caused by imperfectly prepared object to the acquisition device such as staining inhomogeneity [11]. Shading correction methods are basically categorized as *prospective* and *retrospective*, depending upon the degree of access to the available information related to a given image. *Prospective* methods are related to calibration and improvement of image acquisition process. On the other hand, *retrospective* methods solely rely on the information captured in an acquired image. Sometimes, *a priori* information may also be present in case of *retrospective* methods [11]. Since, in this study, we are dealing with microscopic images with no related information about the acquisition process and parameters, only *retrospective* methods will be discussed.

A general assumption usually made on intensity inhomogeneity is that the shading is a smooth, spatially varying function that corrupts image intensities of the imaged objects. In absence of such a shading, image intensities of the imaged objects would be the same irrespective of the location of imaged objects in an acquired image. Various methods have been proposed in the field of *retrospective* shading correction methods using a linear image formation model, consisting of additive and multiplicative shading components in an acquired image [5, 10, 11].

Retrospective methods can be simplified by using only one shading component i.e. additive. *Retrospective* methods are further divided into several approaches such as filtering, surface fitting, segmentation, histograms and others [11]. In case of

filtering, a reliable removal of shading effect from true image data is only possible in the case where the spectra of shading component and true image data are not overlapping. Filtering techniques are more suited for small scale structured images [5]. On the other hand, surface fitting methods are parametric methods approximating smoothly varying image background by parametric surfaces such as polynomials. The optimal parameters for the background can be ascertained by iterative searching in parametric space and optimizing certain criterion such as squared error or other robust distance measure between the parametric background and the acquired image [5].

A brief overview of variety of *retrospective* methods in term of applicability and comparison are given in [5, 6, 10, 11, 12]. However, these methods are limited in terms of well-formulated search space for ascertaining optimal normalization parameters and integration of segmentation results in addition to internal consistency check for the performance of normalization routines. Therefore, we propose a new automatic feedback-driven normalization technique for tuning processing parameters iteratively using fuzzy formulation of *a priori* segmentation knowledge and criteria for normalization evaluation to obtain optimal parameter set. Feedback-driven automatic approaches for segmentation have already been employed in [1, 3, 4, 9] but application to normalization is yet to be explored deeply. Since biomedical image processing is currently a quite challenging domain for image analysis, we chose two datasets of images containing living and dying cells for the evaluation of our normalization technique. Both these datasets were previously used in [4] for automatic tuning of image segmentation parameters in the same fashion. Moreover, the proposed technique could be efficiently used for the fault detection in large datasets. Based on the automatic feedback-based normalization evaluation criteria proposed in this paper, a user can easily discern faulty images in a given image dataset after normalization procedure.

This paper is organized as follows: The methodology of image normalization, subsequent parameter calculation and evaluation criteria are given in Section 2. The results of our proposed technique are given in Section 3 using two biological image datasets followed by conclusions given in Section 4.

2 Methods

In this study, we used a surface fitting *retrospective* shading correction method with parametric fitting to a surface described by a parabolic polynomial. The relation between an acquired image $I_A(x,y)$ and the true shading-free image $I_{SF}(x,y)$ is then described by:

$$I_A(x,y) = I_{SF}(x,y) \cdot S_m(x,y) + S_a(x,y) \tag{1}$$

where, $S_a(x,y)$ and $S_m(x,y)$ denote additive and multiplicative shading components respectively as a function of spatial pixel locations in $I_A(x,y)$. The components $S_a(x,y)$ and $S_m(x,y)$ account for the brightness adjustment and global contrast respectively in Eq. (1). In case of fluorescence microscopy, the acquired images are always distorted with additive shading noise. For the sake of simplicity, only $S_a(x,y)$

was considered, accounting only for the brightness correction in the normalization procedure, such that:

$$I_A(x,y) = I_{SF}(x,y) + S_a(x,y). \tag{2}$$

The proposed scheme for the feedback-based normalization technique is shown in Fig. 1. It includes the normalization of an input grayscale image by fitting the pixel values of $I_A(x,y)$ to $S_a(x,y)$. Image normalization is then performed for different parameter combinations based on parameters obtained from the fitting. Normalization results are evaluated for internal consistency of normalization procedure iteratively to adopt an optimal parameter set $\hat{\mathbf{a}}_{opt}$. Automatic segmentation based on normalized image using $\hat{\mathbf{a}}_{opt}$ is also done iteratively to adopt optimal segmentation parameter set $\mathbf{p}_{opt}$ in an iterative fashion. Optimum segmentation is performed based on this $\mathbf{p}_{opt}$ obtained from optimally normalized image using $\hat{\mathbf{a}}_{opt}$.

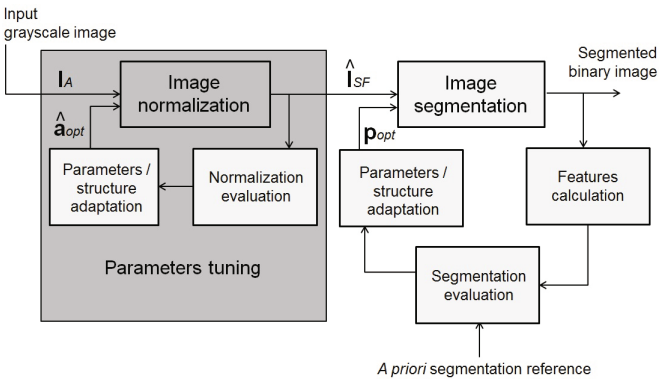


Fig. 1 Employed feedback-based automatic normalization and segmentation scheme

The aim to introduce automatic normalization was to improve automatic feedback-based segmentation results based on *a priori* reference knowledge about cells to be found. *A priori* knowledge included area, intensity and roundness factor (i.e. ratio of major cell axis to sum of major and minor axes) of normal cells in addition to the total number of cell count in an image. *A priori* segmentation reference was given based on manual labeling of the cells in image dataset discussed in Section 3. The automatic feedback-based segmentation is fully discussed in our previous work [4].

Image normalization: As an example, a free exponent polynomial was used for our feedback-based normalization routine. Since we are just estimating the shading-free image denoted as $\hat{\mathbf{I}}_{SF}$, the employed polynomial function denoted as $\hat{S}_a(x,y)$ used for estimation is given as:

$$\hat{S}_a(x,y) = \hat{a}_0 + \hat{a}_1 x + \hat{a}_2 y + \hat{a}_3 xy + \hat{a}_4 x^{\hat{a}_6} + \hat{a}_5 y^{\hat{a}_7}. \tag{3}$$

In order to induce better clarity in mathematical representation, the polynomial function and the acquired image denoted in matrix form as $\hat{\mathbf{S}}_a$ and $\mathbf{I}_A$ respectively, can be written in vectorized form as:

$$\hat{\mathbf{s}}_a^* = \mathbf{X} \cdot \hat{\mathbf{a}}, \tag{4}$$

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1^{\hat{a}_6} & 1^{\hat{a}_7} \\ 1 & 2 & 1 & 2 & 2^{\hat{a}_6} & 1^{\hat{a}_7} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & m & 1 & m & m^{\hat{a}_6} & 1^{\hat{a}_7} \\ 1 & 1 & 2 & 2 & 1^{\hat{a}_6} & 2^{\hat{a}_7} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & m & n & mn & m^{\hat{a}_6} & n^{\hat{a}_7} \end{bmatrix}, \quad \hat{\mathbf{a}} = \begin{bmatrix} \hat{a}_0 \\ \hat{a}_1 \\ \hat{a}_2 \\ \hat{a}_3 \\ \hat{a}_4 \\ \hat{a}_5 \end{bmatrix}, \quad \mathbf{i}_A^* = \begin{bmatrix} I_A(x_1, y_1) \\ . \\ I_A(x_m, y_1) \\ I_A(x_1, y_2) \\ . \\ I_A(x_m, y_2) \\ . \\ . \\ I_A(x_m, y_n) \end{bmatrix}, \quad \hat{\mathbf{a}}_{total} = \begin{bmatrix} \hat{a}_0 \\ \hat{a}_1 \\ \hat{a}_2 \\ \hat{a}_3 \\ \hat{a}_4 \\ \hat{a}_5 \\ \hat{a}_6 \\ \hat{a}_7 \end{bmatrix}. \tag{5}$$

In Eq (4), $\hat{\mathbf{s}}_a^*$ is the vectorized form of $\hat{\mathbf{S}}_a$. In Eq (5), m and n refer to the number of rows and columns of $\mathbf{I}_A$ respectively. In Eq. (4), the dimensions of $\hat{\mathbf{s}}_a^*$ are $mn \times 1$ whereas the dimensions of $\mathbf{X}$ and $\hat{\mathbf{a}}$ are $mn \times 6$ and 6×1 respectively. The vector $\mathbf{i}_A^*$ in Eq. (5) also has dimensions of $mn \times 1$. Since the analytical solution of our problem including $\hat{a}_6$ and $\hat{a}_7$ is not possible, numerical optimization in order to achieve parameters for best fit of $I_A(x,y)$ to $\hat{S}_a(x,y)$ was performed using analytical least squares minimization for $\hat{\mathbf{a}}$. Our normalization routine was divided into two steps:

Step 1 - Initial estimate: Firstly, the least square regression problem is to find:

$$Q_{S1} = \min_{\hat{\mathbf{a}}} \frac{1}{2} ||\mathbf{X} \cdot \hat{\mathbf{a}} - \mathbf{i}_A^*||^2 \tag{6}$$

such that, the optimization initially is only performed setting the exponents $\hat{a}_6$ and $\hat{a}_7$ equal to 2 and finding the optimal parameter values with respect to $\hat{\mathbf{a}}$ to yield an initial estimate of the search space used in the succeeding step of our feedback-based normalization routine.

Step 2 - Search space exploration: The feedback-based normalization is then continued using search space spanned by varying $\hat{a}_6$ and $\hat{a}_7$ around initial estimates to find the optimal combination of $\hat{\mathbf{a}}$, $\hat{a}_6$ and $\hat{a}_7$ that yields the best parameter set to perform the optimum image normalization. The least squares problem for automatic tuning of normalization routine was to find:

$$Q_{S2} = \min_{\hat{a}_6, \hat{a}_7} \frac{1}{2} ||Q_{S1}||^2. \tag{7}$$

The calculation of criterion (7) was based on least square minimization in order to fit $\hat{S}_a(x,y)$ to pixel values of $I_A(x,y)$. Pseudo inverse of the system matrix $\mathbf{X}$ was used to calculate $\hat{\mathbf{a}}$ for given values of $\hat{a}_6$ and $\hat{a}_7$ as:

$$\hat{\mathbf{a}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{i}_A^* \tag{8}$$

to find an optimal solution for the inner criterion Q_{S1}. Consequently, $\hat{\mathbf{I}}_{SF}$ is calculated using its vectorized form $\hat{\mathbf{i}}_{SF}^*$ as:

$$\hat{\mathbf{i}}_{SF}^* = \mathbf{i}_A^* - \mathbf{X}\cdot\hat{\mathbf{a}}, \tag{9}$$

where the dimensions of $\hat{\mathbf{i}}_{SF}^*$ are same as that of both $\mathbf{i}_A^*$ and $\mathbf{X}\cdot\hat{\mathbf{a}}$. The range for $\hat{a}_6$ and $\hat{a}_7$ was kept closely around quadratic between 1.9 to 2.1 based on initial estimate in order to find best exponents yielding optimal image normalization. The employed algorithm also has the possibility to enhance its running time efficiency by downscaling the resolution of $\mathbf{I}_A$ by a certain factor f such that newer $\mathbf{i}_A^*$ has lesser overall elements by factor f. This reduces number of calculations and consequently speeds up the automatic normalization routine.

Normalization evaluation: Generally, a normalization outcome is evaluated with respect to two different criteria i.e. check of internal consistency of spatial image intensities and improvement of image post-processing results based on normalized image. A variety of metrics can be used as a quality measure for normalization procedure with respect to given *a priori* knowledge. In this paper, we used one criterion for the residual root mean square error $Q_{norm,1}$ of acquired image $\mathbf{I}_A$ to the fitted function $\hat{S}_a(x,y)$ normalized by the range of given image intensities and is defined as:

$$Q_{norm,1} = \frac{\sqrt{\frac{1}{m\times n}\sum_{i=1}^{m}\sum_{j=1}^{n}(I_A(x_i,y_j) - \hat{S}_a(x_i,y_j))^2}}{|max\{I_A\} - min\{I_A\}|}. \tag{10}$$

The idea behind using criterion (10) is to calculate the deviation (i.e. residual shading after fitting $\mathbf{I}_A$ to $\hat{S}_a(x,y)$) of $\mathbf{I}_A$ from a uniform background which we intend to obtain in normalized $\hat{\mathbf{I}}_{SF}$, provided that the image has shading noise of the form given in Eq. (3). However, we do not estimate the background here, instead fitting is done based on both back- and foreground intensity information. This can be dealt with by using a sliding filter to spatially inhibit the effect of high brightness in any image. Nevertheless, an ideally normalized image would consist of nearly a uniform background just containing the information about segments (e.g. cells etc.) to be found. This would be indicated by smaller values (i.e. close to zero) of criterion (10). We used another criterion $Q_{norm,2}$, given in Eq. (11), based on the normalized sum of absolute difference in median image intensity values over rows and columns of our estimated shading-corrected image $\hat{\mathbf{I}}_{SF}$ by applying q % percentiles on pixel median values. It is used to evaluate the decrease in spatial intensity value differences of rows and columns as corrected by our shading correction algorithm. In ideal case, for a fairly good segment distribution and shading trends having middle section brighter than corners with no segment having area greater than 50 % of the pixels, such a criterion should yield values closer to 0. The percentile operator using q is denoted as $pr_q\,(\cdot)$ for lower *qth* percentile and $pr_{100\text{-}q}\,(\cdot)$ for the upper *(100-q) th* percentile in Eqs. (11) and (12).

$$Q_{norm,2} = \frac{\left|pr_{100-q}(\mathbf{x}_{med}) - pr_q(\mathbf{x}_{med})\right| + \left|pr_{100-q}(\mathbf{y}_{med}) - pr_q(\mathbf{y}_{med})\right|}{2 \times max\{|t|, 1\}}, \quad (11)$$

$$t = max\{pr_{100-q}(\mathbf{x}_{med}), pr_{100-q}(\mathbf{y}_{med})\} - min\{pr_q(\mathbf{x}_{med}), pr_q(\mathbf{y}_{med})\}, \quad (12)$$

$$\mathbf{x}_{med} = \begin{bmatrix} x_{med,1} \\ . \\ . \\ x_{med,m} \end{bmatrix}, \quad \mathbf{y}_{med} = \begin{bmatrix} y_{med,1} \\ . \\ . \\ y_{med,n} \end{bmatrix}, \quad (13)$$

$$x_{med,i} = \underset{j=1\ldots n}{med} \{\hat{I}_{SF}(x_i, y_j)\} \quad \forall \quad i = 1\ldots m, \quad (14)$$

$$y_{med,j} = \underset{i=1\ldots m}{med} \{\hat{I}_{SF}(x_i, y_j)\} \quad \forall \quad j = 1\ldots n, \quad (15)$$

where, *med* $\{\cdot\}$, *max* $\{\cdot\}$ and *min* $\{\cdot\}$ in Eq. (11), (12), (13), (14) and (15) represent median, maximum and minimum operators respectively. In criterion (11), 2 in the denominator is used to average the effect of variations in percentile image intensity values along columns and rows. The criterion (11) will ideally yield values closer to 0 in case of minimum spatial intensity variations along the row and columns, which would indicate median values along all row and columns are closer to each other. This would tend to eliminate the high spatial variations in intensity, imparting a uniform background which is our desired goal in the shading correction.

We propose a fuzzy formulation of the error measures described in Eq. (10) and (11) to grasp the parametric effect in a more intuitive way. Spline-based (a.k.a *z-shaped*) fuzzy membership functions were employed with two parameters i.e. a and b defining the maximum and minimum x-values of criteria respectively. Fuzzy memberships for the criteria (10) and (11) are denoted as μ_1 and μ_2 respectively. It is reasonable to calculate a criterion $Q_{fuzz,norm}$ based on a product of μ_1 and μ_2 since fulfillment of both criteria for each evaluation is essential and complete absence of any (i.e. $\mu_1 = 0$, $\mu_2 = 0$) should render $Q_{fuzz,norm}$ zero. Therefore, a criterion:

$$Q_{fuzz,norm}(\hat{\mathbf{a}}_{total}) = \mu_1(\hat{\mathbf{a}}_{total}) \cdot \mu_2(\hat{\mathbf{a}}_{total}) \quad (16)$$

based on aforementioned logic, is introduced to express the internal consistency of automatic image normalization.

Parameters/Structure adaptation: The criterion (16) needs to be maximized in order to obtain:

$$\hat{\mathbf{a}}_{opt,fuzzy,norm} = \arg\max_{\hat{\mathbf{a}}_{total}} Q_{fuzz,norm}(\hat{\mathbf{a}}_{total}). \quad (17)$$

In this paper, $\hat{\mathbf{a}}_{opt,fuzzy,norm}$ was computed based on exhaustive enumeration. However, more sophisticated optimum search methods such as genetic algorithms, constraint optimization etc. could be used as well.

Image segmentation: The automatic feedback-driven image segmentation is performed on $\hat{\mathbf{I}}_{SF}$ based on $\hat{\mathbf{a}}$. The image segmentation is performed in the same way as shown in Fig. 1. The optimal parameter set $\mathbf{p}_{opt}$ is adopted based on:

$$\mathbf{p}_{opt} = \arg\max_{\mathbf{p}} Q_{fuzz,seg}(\mathbf{p}), \tag{18}$$

$$Q_{fuzz,seg}(\mathbf{p}) = \mu_c(\mathbf{p}) \cdot \frac{1}{n(\mathbf{p})} \sum_{i=1}^{n(\mathbf{p})} \left(\prod_{j=1}^{m} \mu_{ij}(\mathbf{p})\right). \tag{19}$$

Trapezoidal membership functions $\mu_j(x_j)$ with four parameters (i.e. *a-d* defining x-values of edges of a trapezoid) were used to formulate reference features. Fuzzy membership μ of each segment i for each feature j is denoted as μ_{ij} in Eq. (19) and m denotes the number of considered features. Moreover, the total number n of expected segments in an image was also formulated as a feature of a single image segmentation process using a trapezoidal fuzzy membership function denoted as μ_c in Eq. (19). Elaborate discussion can further be read in [4].

3 Results

Benchmark dataset Human HT29 Colon Cancer 1: This benchmark dataset was published in the Broad Bioimage Benchmark Collection[1]. It contains microscopic images (showing cells) B_k where $k = 1...6$, shown in Fig. 2. The ground truth for B_k was only the average total number n_{ref} of cells present in each image based on manual counting of two observers. For cell detection and counting, the benchmark has to be evaluated by:

$$\sigma_{GD} = \frac{\|n - n_{ref}\|}{n_{ref}} \tag{20}$$

where n is the number of detected cells and σ_{GD} is the deviation from the ground truth in each image. By using $Q_{fuzz,seg} = 1 - \min(\sigma_{GD}, 1)$, to transfer this given criterion into a fuzzy evaluation, $\mathbf{p}_{opt}$ was adopted by (20) based on (17). In addition, a feed-forward automatic segmentation technique proposed by Otsu [8] was applied resulting in a threshold t. However, the results could not be directly compared based just on the consideration of the total number of segments n because in Otsu's method some segments can be considered as insignificant due to their size being considerably smaller than the normal segments. To solve this problem, an image opening was applied in case of Otsu's method in order to remove erroneous small segments (opening filter size $s = 3$). With the addition of an image opening operation, the parameter vector for Otsu's method is described as $\mathbf{p}_{Otsu} = (s,t)^T$. Using values of s larger than 3 causes more deviation from the ground truth and were therefore avoided in case of Otsu's method.

The segmentation results obtained from our feedback-based normalization were improved in comparison to the results obtained from automatic feedback-based

[1] `http://www.broadinstitute.org/bbbc/`

Table 1 Reference cells detection and cell count results from B_k using our method with and without automatic normalization in comparison to Otsu's method using $s = 3$

	Our method (w/o norm.)		**Our method (with norm.)**		**Otsu's method ($s = 3$)**	
Images	σ_{GD} (%)	$x_{1,mean}$	σ_{GD} (%)	$x_{1,mean}$	σ_{GD} (%)	$x_{1,mean}$
B_1	12	110	13	114	8	101
B_2	15	129	12	108	16	97
B_3	18	113	14	105	18	99
B_4	15	99	15	112	12	94
B_5	20	120	18	117	23	96
B_6	11	121	11	128	14	96
μ	**15**	**115**	**14**	**114**	**15**	**97**

segmentation without using normalization. This can be seen in terms of σ_{GD} and mean cell area $x_{1,mean}$ in pixels which are aggregated using mean value μ in Tab. 1. The value of μ of σ_{GD} (i.e. 14) for our automatic segmentation based on normalization scheme shows that greater number of cells were detected compared to μ of σ_{GD} (i.e. 15) obtained without using automatic normalization procedure. This difference of 1 is significant in μ of σ_{GD} since slightly more cells were detected separately from each other which were otherwise merged into each other in high shading regions of the image (see Fig. 2). The mean cell area too was not affected adversely, the direct relevance or comparison of which is not stated in *a priori* reference of B_k. Moreover, comparisons to results obtained by Otsu's feed-forward image segmentation method were thoroughly discussed in [4]. The plausibility of visual results with respect to human observation was slightly improved when using our proposed method. This is indicated in B_2 by a fine delineation of cells lying very close to each other. Since the shading effect around cells has been minimized in our normalization technique, more cells were detected in the whole dataset B_k. This effect is demonstrated in Fig. 2 (image on the right, section on middle top), where the outer boundaries of detected cells were seen to be hardly touching each other.

Therefore, it can be inferred that by using our normalization method for automatic image segmentation, not only the detection but the counting of the cells was also improved in numbers compared to our previous automatic image segmentation method without normalization and Otsu's method. The mean cell area was also kept intact and better cell delineation was achieved compared to work done in [4]. All parameters for normalization and segmentation were also selected automatically as opposed to manual filter size selection in case of Otsu's method.

Cell detection based on an heterogeneous cell dataset: A biological dataset with images P_l where $l = 1...4$ was used as shown in Fig. 3(a). This dataset consists of images showing human lung cells (A549) treated with the anticancer drug cis-platin for 24 hours and representative images were acquired as described previously in [2]. The parameter vector to be optimized in this case is $\mathbf{p} = (r,s,t)^T$, where r, s and t represent convolution square matrix size, image opening disc size and brightness threshold respectively using (18) and (19) with $m = 3$. The shading effect, using P_1

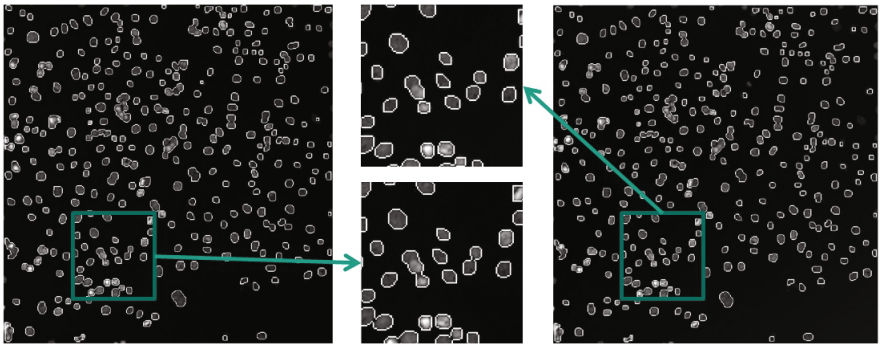
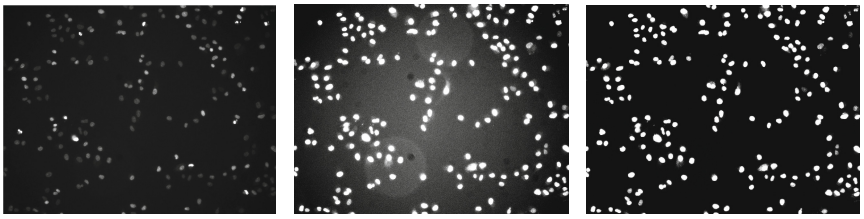


Fig. 2 Comparison of visual results between automatic segmentation without normalization (left) with one using normalization (right) with sectioned images in the middle

having numerous cells in different states, is illustrated in Fig. 3(a). The image suffers from an inconsistent background illumination since the background in corners is darker than in the middle of the image as shown in Fig. 3(b) using high contrast by applying 5 % percentiles. Therefore, shading correction is imperative in order to disentangle image from such variations concealing original intensity information in an image.



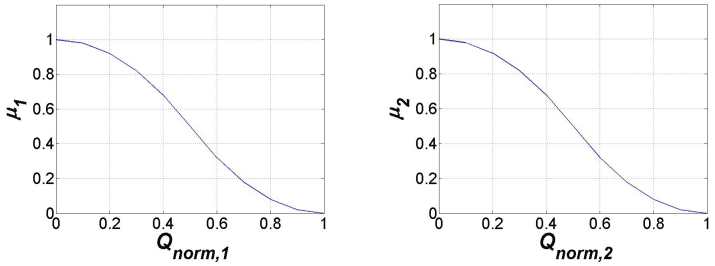
(a) Original grayscale image (b) Contrast-stretched image using 5 % percentiles (c) Contrast-stretched normalized image using 5 % percentiles

Fig. 3 Shading effect and its automatic correction using P_1

Normalization of P_1 image using $\hat{\mathbf{a}}_{opt,fuzzy,norm}$ is done and resulting image is shown in Fig. 3(c). Using high contrast by applying 5 % percentiles, it is clear that shading effects are highly reduced. The spline-shaped fuzzy membership functions were used (see Fig. 4) for the evaluation of our normalization criteria. The results are presented in Tab. 2. It can be seen from Tab. 2 that segmentation criterion $Q_{fuzz,seg}$ was improved in the case of using normalization before automatic segmentation. It demonstrates the beneficial effect of using a shading correction procedure. Uniform backgrounds were achieved (see $Q_{fuzz,norm}$ in Tab. 2) using our automatic normalization technique, whereas in original images of P_1, criterion (11) yielded undesirably

low values. In Tab. 2, $Q_{fuzz,seg}$ represents the total quality of the automatic segmentation procedure, both for original and normalized images. This methodology can be effectively adapted to discard outlier images (i.e. having very low criterion values for $Q_{fuzz,norm}$ and $Q_{fuzz,seg}$) in the whole dataset.

This work was done in continuation to our previous work described in [4]. All algorithms are implemented in MATLAB using the Image Processing Toolbox and the open source Gait-CAD Toolbox [7] for data mining.



(a) $Q_{norm,1}$ vs. μ_1 with (a,b) = (0 1) (b) $Q_{norm,2}$ vs. μ_2 with (a,b) = (0 1)

Fig. 4 Fuzzy spline-shaped membership functions μ_1 and μ_2 for criteria (10) and (11)

Table 2 Comparison of segmentation results with/without automatic normalization of P_1

	Original images		**Normalized images**	
Images	$Q_{fuzz,norm}$	$Q_{fuzz,seg}$	$Q_{fuzz,norm}$	$Q_{fuzz,seg}$
P_1	0.15	0.73	1	0.75
P_2	0.07	0.76	1	0.76
P_3	0.08	0.77	1	0.78
P_4	0.11	0.69	1	0.73

4 Conclusions

It was shown from our results that feedback-oriented normalization algorithms using fuzzy criteria have the capability to fulfill the goals of image segmentation using a human reference. The presented scheme was able to produce good results, using two biological image datasets, in terms of number and quality of segments found based on automatic normalization.

In the future, exhaustive enumeration for finding the optimal parameter set would be replaced by nonlinear optimization. Moreover, new benchmarks would be tested for the reliability of performed normalization and multiplicative shading component would also be accounted for in shading correction based on *a priori* knowledge.

Acknowledgements. We express our gratitude to DAAD and BioInterfaces program of the Helmholtz Association for funding this research work.

References

[1] Beller, M., Stotzka, R., Müller, T.: Application of an interactive feature-driven segmentation. Biomedizinische Technik 49(E2), 210–211 (2004)
[2] Donauer, J., Schreck, I., Liebel, U., Weiss, C.: Role and interaction of p53, bax and the stress-activated protein kinases p38 and jnk in benzo(a)pyrene-diolepoxide induced apoptosis in human colon carcinoma cells. Archives of Toxicology 86(2), 329–337 (2012)
[3] Farmer, M., Jain, A.: A wrapper-based approach in image segmentation and classification. IEEE Transactions on Image Processing 14(12), 2060–2072 (2005)
[4] Khan, A., Reischl, M., Schweitzer, B., Weiss, C., Mikut, R.: Automatic tuning of image segmentation routines by means of fuzzy feature evaluation. In: Proc., 6th International Conference on Soft Methods in Probability and Statistics, Konstanz, Germany. SCI. Springer (accepted paper, 2012)
[5] Likar, B., Maintz, J., Viergever, M., Pernus, F., et al.: Retrospective shading correction based on entropy minimization. Journal of Microscopy 197(3), 285–295 (2000)
[6] Mai, F., Chang, C., Liu, W., Xu, W., Hung, Y.: Segmentation-based retrospective shading correction in fluorescence microscopy E. coli images for quantitative analysis. In: Proc., International Symposium on Multispectral Image Processing and Pattern Recognition, vol 7498, pp. 74,983O–1 (2009)
[7] Mikut, R., Burmeister, O., Braun, S., Reischl, M.: The open source Matlab toolbox GaitCAD and its application to bioelectric signal processing. In: Proc., DGBMT-Workshop Biosignalverarbeitung, Potsdam, pp. 109–111 (2008)
[8] Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man and Cybernetics 9, 62–66 (1979)
[9] Reischl, M., Alshut, R., Mikut, R.: On robust feature extraction and classification of inhomogeneous data sets. In: Proc., 20. Workshop Computational Intelligence, Forschungszentrum Karlsruhe, pp. 124–143 (2010)
[10] Tomaževič, D., Likar, B., Pernuš, F.: Comparative evaluation of retrospective shading correction methods. Journal of Microscopy 208(3), 212–223 (2002)
[11] Vovk, U., Pernus, F., Likar, B.: A review of methods for correction of intensity inhomogeneity in MRI. IEEE Transactions on Medical Imaging 26(3), 405–421 (2007)
[12] Zwier, J., Van Rooij, G., Hofstraat, J., Brakenhoff, G.: Image calibration in fluorescence microscopy. Journal of Microscopy 216(1), 15–24 (2004)

Part IV
Intelligent Data Analysis

Exploring Time Series of Patterns: Guided Drill-Down in Hierarchies Using Change Mining on Frequent Item Sets

Mirko Böttcher and Martin Spott

Abstract. In the past years pattern detection has gained in importance for many companies. As the volume of collected data increases so does typically the number of found patterns. To cope with this problem different interestingness measures for patterns have been proposed. Unfortunately, their usefulness turns out to be limited in practical applications. To address this problem, we propose a technique for a guided, visual exploration of patterns rather than presenting analysts with static ordered lists of patterns. Specifically, we focus on a method to guide drill-downs into hierarchical attributes, where we make use of change mining on frequent item sets for pattern discovery.

1 Introduction

Companies are faced with a market environment that changes faster than ever. Product life cycles are getting shorter, and prices and therefore profit margins shrink due to harder competition. At the same time customers demand higher service levels. To tackle the problem, organisations collect vast amounts of data about customers, internal processes and external influences at increasing rates in order to make smart business decisions. Nevertheless, they still fail to unfold the full potential of the data for their decision making. The resulting lack of information often leads to suboptimal decisions.

Pattern detection is at the heart of most data analysis activities. However, analysts usually only find the patterns they are looking for. Typically, the analysis process consists of formulating a hypothesis based on domain knowledge and testing its

Mirko Böttcher
University of Magdeburg, Faculty of Computer Science, 39106 Magdeburg, Germany
e-mail: miboettc@iws.cs.uni-magdeburg.de

Martin Spott
BT, Research and Technology, Ipswich, IP5 3RE, UK
e-mail: martin.spott@bt.com

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 181–194.
springerlink.com

validity. For instance, if the number of incoming jobs of a service provider is rising over time, then an analyst may test whether this was driven by an increasing demand for certain products, in certain areas, by specific types of job etc. The number of hypotheses is however limited by the analysts' time and imagination. To make things worse, some scientists like pharmacologists even argue that they do not know what they are looking for, that they will only know when they see it. This suggests that we must look for a clever way to present potentially interesting patterns to users rather than expect them to come up with hypotheses.

Unfortunately, machines still struggle to evaluate the interestingness of found patterns, to automatically make decisions and trigger actions based on them, mainly due to the lack of domain knowledge. For that reason, we are interested in exploratory data analysis, where machines focus on the mechanical part of analysing large amounts of data and only guide the analysts' exploration of the results through interactive visualisations. The analysts can then include domain knowledge, trigger further analysis by the machine and make decisions based on the results.

One core question is how to hint an expert at the most relevant patterns. While it is very challenging to design an algorithmic method to assess the interestingness of a pattern, it is astonishingly simple for us humans to decide what is relevant to us and what is not. One of the clues to how humans judge the interestingness of an object is that they take its past and how it changes into account. For instance, when investing in stocks or buying expensive consumer goods we do not only look at the current price but also how it developed over the last couple of months. When we like to place a bet we do not only look at how a team scored last weekend but during the whole season.

For a business change can mean a risk (like a shrinking subgroup of target customers) or an opportunity (like an evolving market niche). In either case, the business has to detect the change in order to survive or to win. In some business domains the value of information about change as a key enabler for pro-active decision-making has been known for a long time. For example, stock traders aim to optimise buy and sell decisions by analysing stock price behaviour over time. Moreover, many data collected are already time-stamped. In fact, the time dimension is the one dimension which is present in every data warehouse [8].

In recent years there has been an increasing research interest in methods which aim at analysing the changes within a domain by describing and modeling how the results of data mining—models and patterns—evolve over time. The term change mining has been coined as an umbrella term for such methods [5]. Change mining approaches have been proposed for a variety of patterns and models (see [4]). Many studies focus on analysing change in the context of item sets, not only because item sets are rather comprehensible but also because their evolution can be represented in a convenient and interpretable way.

In a previous publication we reported an application of change mining for item sets in the context of detecting interesting customer segments from data collected by a telecommunications provider [7]. Here, we describe an application which extends this prior work by two aspects: the possibility to use a machine-guided drill-down to find aspects of a domain that change in an interesting way and the

incorporation of attribute hierarchies as they are commonly found as part of corporate data warehouses.

Similar hierarchies as the ones described in this publication are found in OLAP (Online Analytical Processing) tools where a target variable, such as the revenue, is aggregated at various level of detail, for instance, first on country level, then on state level and then on town level. Change explanation methods for OLAP hierarchies have been, for example, proposed by [11] and [1]. In contrast to our approach they only account for two time periods, and they aim towards finding the most drastic differences between the latter. An extension to more than two periods is not straightforward.

This paper is structured as follows. Section 2 describes our data and our analysis task in more detail. Section 3 then continues with formalising this problem by introducing notation to describe attribute hierarchies and the temporal dimension. In particular it introduces histories of weight aggregates as a general way for expressing the temporal development not only of counts (volumes), but also of prices, profits, costs etc. and shows how those trends are identified which appear to contradict more general trends, as defined by the attribute hierarchy. In the following Section 4 we show how this work relates to and is embedded into our previous work on change mining. Section 5 describes an application of the techniques.

2 Problem Description

For illustration, we will use the following exemplary problem throughout the paper, which is very common across different industry sectors. Assume, a service provider receives jobs such as orders or fault reports from customers. Every job has attributes such as time stamp (when the job came in, or when it has been completed), type of job, location, type of customer, product and service level. Such attributes may differ slightly between industries, but at their core they can be assumed to be typical.

The organisation needs to understand, how the number of jobs develops over time for different attribute-value combinations such that new trends in particular segments can be spotted early. In this way, problems can be anticipated and resolved before they escalate. Furthermore, such trends can be used for forecasting and planning.

Most organisations monitor trends only at a global level. For instance, they would only look at the development of the overall number of jobs rather than all the different types of jobs in all different locations for all customer groups etc. Such an approach is reasonable, since the number of different attribute-value combinations grows exponentially with the number of attributes and can be very high—in some real data sets tens of thousands different combinations. If a high-level trend has been detected, the root cause needs to be found. Analysts will try to figure out if the trend is local, i.e. only for a certain type of job, product, location etc. Since the number of different combinations is usually too high to test all of them manually, we are looking to automate the procedure and visualise the results for a guided exploration.

We assume that some attributes are hierarchical, i.e. they can be further broken down into attributes with a higher level of detail. Fig. 1 shows—in partial—a hierarchy for the attribute *location*. Starting at UK level, each level below introduces an attribute which divides the country up into disjoint, more fine-grained parts. For *location* these attributes may be *region*, *county*, *borough*, and so on, until *post code*. For now, we stipulate that every element in the hierarchy has exactly one parent element, i.e. hierarchies can be represented by trees. Additionally we require that the hierarchy is complete in the sense that it contains all children of a node. For instance, for *region* = *South*, we require all counties in the South of the UK as children. We will discuss completeness more formally later on. If a hierarchy is not complete, we can simply add a virtual value *other* covering for the missing values. Neither of the two assumptions is severe, since almost all hierarchies in our real-world data sets follow this schema.

Attributes with no hierarchy, i.e. just a flat set of values, can be given a value *all* as a virtual root node. *all* is equivalent to the set of all values, i.e. the attribute domain. In that way, every attribute can be made hierarchical for consistency.

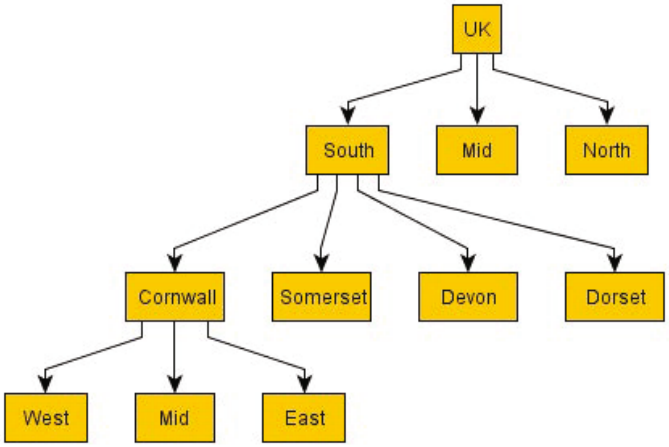


Fig. 1 Partial hierarchy of the attribute *location*

The idea of the proposed exploration approach is to start with a high-level view of trends in the data, such as the overall trend for the UK and then guide the user in drilling down into attributes. For instance, if a trend is observed for the South of the UK but the same trend occurs in all its counties, a drill-down into location will not reveal more information. The other way around, if the trend prevalently occurs for a certain service, then the analyst should be made aware, eventually suggesting him to drill down in this direction.

3 Formalisation

In this section we formalise the previously sketched approach in order to yield a notion of homogeneity within hierarchies which builds upon time series of arbitrary weights.

3.1 Data and Attributes

We assume relational data in form of records (table rows) $r \in \mathscr{D}$, $\mathscr{D}$ being the set of all records, with attributes (table columns) $A \in \mathscr{A}$, $\mathscr{A}$ being the set of all attributes. As outlined above, a record may relate to an order that has been placed and the attributes represent further information about the order such as product, location or service level. Every record has three additional attributes:

- an identifier $id \in \mathscr{I}\mathscr{D}$, e.g. an order number, which is not necessarily unique (the same order may have several records reflecting different stages of the process)
- a time stamp $t \in \mathscr{T}$, e.g. the time when the order has been placed, thereby $\mathscr{T}$ being the overall time period that is the minimum time span that covers all records of $\mathscr{D}$.
- a weight $w \in \mathbb{R}$. This could be as trivial as being the constant 1 as a count, or alternatively represent costs, profit, price etc. associated with the record

Altogether a record can be characterised by a tuple $(id,t,w,a_1,\ldots a_n) = (id,t,w,\mathbf{a})$.

We further assume that each record is described by attributes that are leaves in the hierarchy. The hierarchy itself is part of the corporate background knowledge and typically modelled as dimensions in data warehouses. To pick up on the location example, each record originally contains the post code. Hierarchies which map post codes to a less detailed location description are virtually available to everyone. By looking up the value of the respective county attribute in such a hierarchy and replacing the post code attribute with it, the record is transformed into one that describes the location with less detail.

In an attribute hierarchy we assume that each attribute has exactly one attribute as a direct generalisation, called the parent. This way, the attributes within a hierarchy can be ordered by increasing degree of generalisation. To navigate within a hierarchy we define the function $\mathrm{p}(A)$ producing the parent of A (thereby defining the root node as its own parent). Further, we define $\mathrm{val}_\mathrm{p}(A,a) = b$ or short $\mathrm{val}_\mathrm{p}(a)$ as the function that maps the attribute value $a \in \mathrm{dom}(A)$ to its corresponding (i.e. more general) attribute value $b \in \mathrm{dom}(\mathrm{p}(A))$.

3.2 Weights and Time

Rather than looking at individual records, analysts will aggregate them over subperiods of time (like weeks or months), grouped by attribute-value combinations. In order to achieve that $\mathscr{T}$ is divided into $n > 1$ non-overlapping periods $T_i \subset \mathscr{T}$, such that the corresponding data subsets $\mathscr{D}_i \subset \mathscr{D}$ each have a size $|\mathscr{D}_i| \gg 1$.

Through aggregation, the id will be lost and the weight requires an aggregation operator $\bigoplus$ like sum, mean, maximum or minimum. Overall, the aggregation can be described as a mapping $\text{agg} : (T, \mathbf{a}) \longmapsto \bigoplus(\{w | (id, t, w, \mathbf{a}) \in (\mathscr{I}\mathscr{D}, T, w, \mathbf{a})\})$.

In this way, we can produce a time series of weight values for every attribute vector $\mathbf{a}$: $w_i(\mathbf{a}) := \text{agg}(T_i, \mathbf{a})$. As mentioned above, the time series may describe how the number of weekly reported jobs develops over time, or their associated revenue, costs etc.

The aggregation operator is semantically bound to the weight of a record and must therefore be used for the aggregation in hierarchical attributes, as well. In other words, the weight of a parent in a hierarchy can be computed by aggregating the weights of its children. Formally, let the value of all attributes but one be fixed. Let $B \in \mathscr{A}$ be the variable attribute, b a value of B and $\{b_c \mid \text{val}_\text{p}(b_c) = b\}$ the values of b's children. Without loss of generality, let us omit the fixed attributes in the following and only write the values of B. An aggregation then yields the parent time series

$$\forall i \; w_i(b) = \bigoplus_{\{b_c \mid \text{val}_\text{p}(b_c) = b\}} w_i(b_c) \tag{1}$$

The equation gives us a necessary condition for the completeness of a hierarchy, in that it must hold for every parent in the hierarchy of every attribute. In case of strictly monotonous operators like the sum, we have equivalence, i.e. a sufficient condition for completeness.

Lemma 1. *Let sum be the weight aggregation operator and all weights be positive. A hierarchy is complete iff for all values b of non-leaf attributes $B \in \mathscr{A}$ holds*

$$\forall i \; w_i(b) = \sum_{\{b_c \mid \text{val}_\text{p}(b_c) = b\}} w_i(b_c).$$

Proof of the lemma is straightforward. The property is central for dealing with proportions of records in hierarchies and defining probability measures.

3.3 Homogeneity of Trends

In order to decide whether drilling down into an attribute will reveal interesting patterns, we introduce the concept of *temporal homogeneity*. In general terms we consider a parent-child relationship in an attribute hierarchy temporally homogeneous, if the associated time series show the same behaviour over time. For the formal definition, we again let the value of all attributes but one be fixed and use the same notation for the variable attribute B as above.

Definition 1. An attribute value b is called **temporally homogeneous with its child** b_c, iff $\exists c \in \mathbb{R} : \; \forall i \; w_i(b) = c w_i(b_c)$.

An attribute value being temporally homogeneous with all its children is equivalent to the property that the proportion of weights $w_i(b_c)$ between the children b_c does not change over time i. This can also be interpreted in terms of probability theory.

Given a complete hierarchy with sum as aggregation operator, we can define a probability measure on attribute-value combinations $\mathbf{a}$ as $P(\mathbf{a}, T_i) := w_i(\mathbf{a})/w_i(\text{root})$ with root being the vector of the root value for every attribute. Homogeneity then means that $P(child|parent, T_i)$ is constant over time, i.e. the probability distribution of the children given the parent does not change.

In practice, the above definition is too strict if the data is noisy. We suggest to use measures to quantify the level of homogeneity or run statistical tests. Table 1 shows the Pearson correlation between the time series of a parent area with the ones of the child areas. The time series describe the volume of jobs (weekly aggregated) over a period of 32 weeks in different areas (location).[1] The Pearson correlation is a measure of linear dependence, ± 1 meaning perfect linear dependence which is equivalent with temporal homogeneity according to Def. 1. Areas 5 and 9 show the lowest levels of correlation with the parent region (low homogeneity) and areas 3 and 8 the highest values. The normalised time series in Fig. 2 illustrate the difference in homogeneity between areas 3 and 9. The time series of area 3 matches the one of the parent region much better than the one of area 9.

The flaws of using the Pearson correlation as a measure for linear dependence are well known (e.g. its sensitivity to outliers), but it nevertheless is a useful measure in practice. Alternatively, statistical tests are described in [6].

4 Change Mining for Item Sets

The technique developed in the previous sections is extremely useful for the assumed application area where analysts monitor how values develop over time at a high level and try to narrow down trends found at a global level to local root causes. In general, interactive data exploration by drill-down is typical for OLAP systems as part of corporate data warehouses and the proposed approach augments such systems in that it offers a guided and more focused analysis.

Table 1 Homogeneity of nine areas with the parent region based on Pearson correlation

Homogeneity	$area_1$	$area_2$	$area_3$	$area_4$	$area_5$	$area_6$	$area_7$	$area_8$	$area_9$
correlation	0.88	0.82	0.94	0.92	0.69	0.83	0.91	0.93	0.76

While such a drill-down is a useful tool for strategic and decision making control on an upper management level, it has shortcomings at the operational level: first, a large number of attributes leads to an explosion in the number of paths an analyst may drill down into, but impossibly can. The discovered changes are thus still biased towards an analyst's preferences, and therefore changes may not be discovered at all. Second, the aforementioned hierarchies do not model strong dependencies between attributes values, as they occur when a certain service is only offered in a particular

[1] Real data from a telecommunications company.

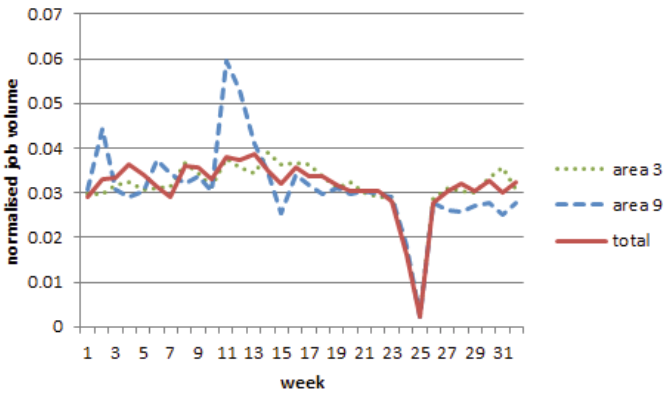


Fig. 2 Normalised time series of the parent region (*total*, in red), area 9 (low level of homogeneity) and area 3 (high level of homogeneity)

region but nowhere else. Since analysts have to drill down one attribute at a time, such dependencies are difficult to discover.

Both issues can be solved by *frequent item set discovery* [2]. Given a data set its goal is to detect all those attribute values which frequently occur together. The advantage of item set discovery is the completeness of its results: it finds the exhaustive set of all patterns which exceed specified thresholds on certain significance metrics. Traditionally, the support of an item set is chosen as such a metric which is defined as the relative frequency of an item set's occurrence in the data. Nevertheless, it is also possible to choose other metrics based on weights assigned to individual items such as profit or price as we entertained in Sect. 3.2 (see [10]). Overall, frequent item set mining provides a rather detailed description of a data set's structure, and thus of the underlying domain.

4.1 Notation

Formally, item set discovery is applied to a data set of *transactions*. Every transaction $\mathscr{R}$ is a subset of a set of items L. A subset $X \subseteq L$ is called *item set*. It is said that a transaction $\mathscr{R}$ *supports* an item set X if $X \subseteq \mathscr{R}$. For reasons of simplicity, we define $XY := X \cup Y$ and $Xy := X \cup \{y\}$.

The statistical significance of an item set X is measured by its *support* $\mathrm{supp}(X)$ which estimates $P(X \subseteq \mathscr{R})$, or short $P(X)$. It is said that an item set is *frequent* if its support is greater than or equal to a user-defined minimum support value $\mathrm{supp}_{\min}$. The *downward closure property* of item sets states that for two item sets $Y \supset X$ the support of X is greater or equal to the one of Y, i.e. $\mathrm{supp}(X) \geq \mathrm{supp}(Y)$.

In our previous work [7] we defined the change of an item set by the change of its support over time. After carrying out frequent item set discovery for each $\mathscr{D}_i$,

$i = 1,\ldots,n$ the support of each item set X is now related to a specific time period T_i. We will indicate this by using the notation $\text{supp}_i(X)$. An item set X which has been discovered in all periods is therefore described by n support values. Imposed by the order of time the values form sequences $(\text{supp}_1(X),\ldots \text{supp}_n(X))$ which are also called *support histories*.

Following the introduction of weights and aggregation functions in Sect. 3.2 we utilise that the item set mining process can also integrate other quantities than support which have a higher practical relevance for industrial applications. Pei et al [10] provide a general framework for this. The above mentioned support threshold $\text{supp}_{\min}$ then becomes a weight threshold $w_{\min}$ and support histories are replaced by *weight histories* $(w_1(X),\ldots w_n(X))$.

4.2 Hierarchical Item Sets and Temporal Homogeneity

A data record r as described in Sect. 3 can be transformed into a transaction $\mathscr{R}$ by encoding every (attribute, attribute value) combination as an item. An item set then describes a conjunction of attributes and their values, and its support represents the relative number of records which satisfy this conjunction. For instance, *region* = *south* is an item, and $\{$*region* = *south*, *service* = *broadband*$\}$ an item set. We define a function item(A,a) which maps an (attribute, attribute value) combination to the corresponding item. In the following, we will omit the attribute name if it is obvious from the context of the value.

The set of *frequent item sets* $I(\mathscr{D}) = \{X \ : \ w(X) \geq w_{\min}\}$ is generated from the data set $\mathscr{D}$ by traversing the power set of L. This step utilises support's *downward-closure property*: $w(X) \geq w(X \cup \{x'\})$ for any item set X and $x' \in L$. This property implies that every proper superset of an infrequent item set is infrequent, too.

When integrating attribute hierarchies into item set mining, it is important to preserve the property of downward closure. This can be achieved by adding all parents of attribute values to item sets up to the value at the top of the hierarchy. For instance, an item set $X = \{$*ipswich*, *broadband*$\}$ would be extended to $X' = X \cup \{$*suffolk*, *east anglia*, *south*$\}$. Since the added items are universally true given the region *ipswich* it is $w(X') = w(X)$. Furthermore, item sets at different levels of the same hierarchy can now be compared. Where item sets $Y = \{$*east anglia*, *broadband*$\}$ and $Z = \{$*south*, *broadband*$\}$ are simply different ($Y \neq Z$), we have inclusion for the extended item sets: $Y' = \{$*east anglia*, *south*, *broadband*$\} \supseteq \{$*south*, *broadband*$\} = Z'$ and $w(Y') \leq w(Z')$.

Having shown the relationship between records and transactions and their time series of weights, we will transfer the concept of a temporally homogeneous parent-child relationship (cf. Sect. 3.3) to item sets.

Definition 2 (Temporally Homogeneous Item Set). Let $XY, X \neq \emptyset$ be item sets and $(w_1(XY),\ldots,\ w_n(XY))$, $(w_1(X),\ldots,\ w_n(X))$ the associated weight histories. The item set X is temporally homogeneous with the item set XY, iff there exists a constant $c \in \mathbb{R}$ such that $w_i(XY) = cw_i(X), i = 1,\ldots,n$.

Since real-world data is typically noisy, the condition for temporal homogeneity should be tested statistically—an appropriate test procedure can be found in [6].

As before, the idea behind the definition is that the history of an item set and hence the item set itself is temporally homogeneous with a more specific item set if their support histories have the same shape apart from a scaling factor c. This can be explained more formally by going back from weight to support histories.[2] The criterion $\mathrm{supp}_i(XY) = c\,\mathrm{supp}_i(X), i = 1,\ldots,n$ used in the definition can be rewritten as $c = \mathrm{supp}_i(XY)/\mathrm{supp}_i(X) = P(XY\,|\,T_i)/P(X\,|\,T_i) = P(Y\,|\,XT_i)$. This means, the conditional probability of Y given X is required to be constant over time, i.e. the fraction of transactions containing Y additionally to X changes in the same proportion as the ones of X.

If the additional item Y is a child attribute value in the hierarchy of one of the values in X, then the two definitions 2 and 1 describe the same phenomenon. In other words, the introduction of item sets allows to generalise the concept of temporal homogeneity of patterns.

Given such time-invariance the history of XY can be inferred if one knows the support history of X (and vice versa). For that reason, XY and X are also called *temporally redundant* [6]. Redundant patterns can be hidden from users in order to reduce the number of potentially interesting patterns. Using set inclusion to form an order on item sets, one can form sets of mutually redundant item sets and reduce the sets to the maximal (most specific) or minimal (most general) elements. In some applications, analysts are more interested in the most specific, in others in the most general item sets. This approach of removing redundant patterns is an extension of *closed item sets* (see [9]) in two ways: first, by adding the dimension time and secondly, through generalising from $c = 1$ in the condition.

If we take hierarchical attributes into account, sets of mutual redundant item sets can be formed along the hierarchies. In the original problem from Sect. 2, analysts are typically interested in the most general item set, since drilling down does not add any information.

5 Putting the Approaches into Practice

In the context of our examples, a data analysis job is typically triggered by detecting trends in the time series either at the highest level of the hierarchies, for instance the overall number of jobs rising in the UK, or for a very specific subset of the data. Since high level trends are usually monitored, they would be automatically detected and lead to further analysis. For this purpose, we have developed interactive visualisation techniques that make use of hierarchies and temporal homogeneity [12]. The idea is to visualise attributes with their hierarchies in a radial tree layout. Given a trend at the root node, the shape of the node indicates for each root attribute, if the underlying children are homogeneous with the parent (circle) or not (triangle). If not, the node can be unfolded one level and the child nodes be revealed. Every child node has a colour and a shape: the colour to indicate its homogeneity with the

[2] As indicated in Sect. 3.3, a similar argument can be constructed with weight histories.

parent (a colour spectrum can be used for the level of homogeneity) and the shape to indicate if the underlying children are homogeneous or not as before.

All hierarchies can be opened up at the same time, however it must be noted that the homogeneity indicators in a hierarchy are always based on the assumption that the other attributes have the root value. This can be changed by selecting values lower down in all hierarchies, which is equivalent to selecting a subset of the data as the new base line for homogeneity tests and the indicators will be recalculated.

In a different mode, homogeneity can be measured between a node and its root node rather than the immediate parent. Fig. 3 shows a visualisation of this mode based on Mike Bostock's Sunburst [3], implemented in d3. This example is essentially a visualisation of Tab. 1, but extended to the volume of jobs of nine regions over 32 weeks with a number of subregions each. The segments of the inner circle represent the regions, the attached segments of the outer circle are the associated subregions. The grey levels indicate the level of homogeneity of a segment with the root, i.e. the overall development of volumes in the UK. Light grey means a high level of homogeneity, medium grey a medium level and black a low level. An analyst will typically focus on the dark segments to look for unexpected developments over time. The sizes of the segments correspond to the volume of jobs in the different regions and help evaluate the gravity of differences. Labels can be added to the segments and subtrees folded and unfolded. Furthermore, other attributes can be included in the circular representation.

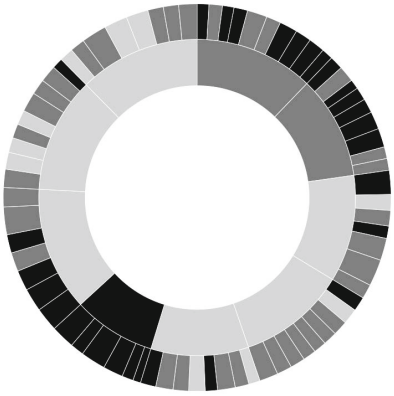


Fig. 3 Circular representation of homogeneity of nine regions (inner circle) and subregions (outer circle) with the root. Light grey indicates a high level of homogeneity, medium grey a medium level and black a low level.

In the second scenario, trends will be detected for very specific subsets of the data set, described by item sets at lower levels of the hierarchy. After having induced item sets with their weight histories and having removed redundant ones, the weight histories can be analysed for trends (upward, downward, stable, spikes etc).

Furthermore, the item sets can be sorted regarding the significance or strength of their trends. Fig. 4 shows a screenshot of the developed tool IDEAL with a list of association rules found, sorted by the strength of their downward trend. The time series of support and confidence of the highlighted rule are displayed in the bottom panel.

Users find patterns with highly significant trends at the top of the sorted list and can then judge, if the found patterns are of interest. If that is the case, the trend of related patterns—item sets that share items—can be compared with the first in order to gain additional information. Furthermore, potentially unrelated item sets that show a related trend can be retrieved for the same reason.

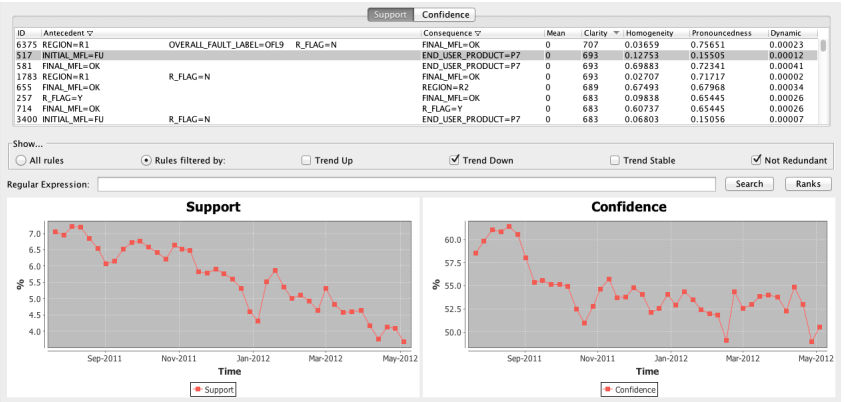


Fig. 4 Screenshot of our tool IDEAL that discovers temporal association rules from data, and analyses and evaluates trends of support and confidence time series

Both approaches have been tested on real-world data in a telecommunications company and have revealed unexpected temporal patterns. Some of them could be explained by internal changes such as new business processes or the replacement of products. However, the tool also discovered changes in customer behaviour that had been missed otherwise.

6 Conclusion

Data is collected by organisations at an increasing rate and at the same time they are struggling to make use of it for decision making. Change Mining is a promising approach to tackle this problem, since it will automatically find interesting patterns in temporal data, analysts may otherwise miss. However, it still suffers from the problem that a lot of potentially interesting item sets will be generated even if methods are employed to remove redundant ones. Interestingness measures which essentially produce an ordered list of item sets can help, but they are not the optimal choice for

business applications because they neglect the interconnectedness of item sets as imposed by, for example, attribute hierarchies.

Drilling down into attribute hierarchies from a top view can overcome this problem, but brings practical problems in domains with a large number of attributes. Since a drill-down can only be done one attribute at a time, too many steps are required before sufficiently small parts of a domain are identified. Here, change mining for item sets comes into play. Due to its strength to automatically analyse the change in arbitrarily small parts of a domain, it can be employed as a look-ahead mechanism for the drill-down by precalculating trends and their homogeneity across a hierarchy.

This paper proposes a way to combine the two approaches in a mathematically consistent way and introduces the concept of temporal homogeneity of hierarchical patterns. We believe that this combination will reveal patterns, analysts would not have found otherwise and at the same time give them a tool to visually explore patterns much quicker than in the past.

Future work includes research in the areas of interactive visualisation of temporal patterns as well as the integration of domain knowledge.

References

[1] Agarwal, D., Barman, D., Gunopulos, D., Young, N.E., Korn, F., Srivastava, D.: Efficient and effective explanation of change in hierarchical summaries. In: KDD 2007: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 6–15. ACM, New York (2007), http://doi.acm.org/10.1145/1281192.1281197

[2] Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 207–216. ACM, Washington D.C. (1993)

[3] Bostock, M.: Sunburst, visualisation example for d3. (2012), http://mbostock.github.com/d3/ex/sunburst.html

[4] Böttcher, M.: Contrast and change mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1(3), 215–230 (2011), doi:10.1002/widm.27

[5] Böttcher, M., Spiliopoulou, M., Höppner, F.: On exploiting the power of time in data mining. SIGKDD Explorations Newsletter 10(2), 3–11 (2008)

[6] Böttcher, M., Spott, M., Kruse, R.: A Condensed Representation of Itemsets for Analyzing Their Evolution over Time. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009. LNCS, vol. 5781, pp. 163–178. Springer, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-04180-8_28

[7] Böttcher, M., Spott, M., Nauck, D., Kruse, R.: Mining changing customer segments in dynamic markets. Expert Systems with Applications 36(1), 155–164 (2009), http://dx.doi.org/10.1016/j.eswa.2007.09.006

[8] Kimball, R.: Data Warehouse Toolkit: Practical Techniques for Building High Dimensional Data Warehouses. John Wiley & Sons (1996)

[9] Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient mining of association rules using closed itemset lattices. Information Systems 24(1), 25–46 (1999)

[10] Pei, J., Han, J., Lakshmanan, L.V.S.: Pushing convertible constraints in frequent itemset mining. Data Mining and Knowledge Discovery 8(3), 227–252 (2004), http://dx.doi.org/10.1023/B:DAMI.0000023674.74932.4c

[11] Sarawagi, S.: Explaining differences in multidimensional aggregates. In: VLDB 1999: Proceedings of the 25th International Conference on Very Large Data Bases, pp. 42–53. Morgan Kaufmann Publishers Inc., San Francisco (1999)

[12] Schmidt, F., Spott, M.: Visualising temporal item sets – guided drill-down with hierarchical attributes. In: Proceedings of Soft Methods in Probability and Statistics, SMPS 2012 (2012)

Enriching Multivariate Temporal Patterns with Context Information to Support Classification

Frank Höppner, Sebastian Peter, and Michael R. Berthold

Abstract. In this paper we consider classification tasks where the class depends on the co-evolution of multiple variables over time, for instance, "if A happens before B and in the meantime we do not observe C, then we have a failure of class X". We present a two-phased approach to derive such patterns from data. In the first step, we seek the most specific pattern that still matches all data from one class and in the second step we constrain the pattern further, such that it discriminates with respect to other classes. While the second step is directly motivated by the classification task, the first step enables the user to better match his or her mental model of the temporal process to the patterns derived by the classifier. The experimental evaluation on the libras dataset has shown that the additional first step not only improves the interpretability, but also the classification results.

1 Introduction

Measuring and recording data is easy and cheap nowadays, but in some applications substantial conclusions can only be drawn if we extend our observations to a certain period of time. An operator who is controlling a chemical production process, a user interacting with a technical device, a medic administering a drug to a patient – in all these cases instantaneous information does not help to differentiate between a successful and a failed process, decide about the ergonomics of a man-machine interface or judge about the chances of patient recovery. It is necessary to observe

Frank Höppner
Ostfalia University of Applied Sciences, Department of Computer Science, 38302 Wolfenbüttel, Germany
e-mail: f.hoeppner@ostfalia.de

Sebastian Peter · Michael R. Berthold
University of Konstanz, Nycomed-Chair for Bioinformatics and Information Mining, 78457 Konstanz, Germany
e-mail: {sebastian.peter,michael.berthold}@uni-konstanz.de

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 195–206.
springerlink.com

multiple attributes over a period of time to derive rules specifying how a class label may depend on the *history* of observations.

Measuring a couple of variables over a period of time turns the classification task into a high-dimensional problem. As classifiers seek for the *best attribute* to discriminate between the classes, they may eventually come up with classification rules that only depend on a few of these attributes. However, users wants to align the findings with their mental model, which is difficult if most of the temporal context is ignored or lost by the classifier. In this paper, we propose to include more background information by means of a temporal outline or sketch, which is then refined by the classifier. This approach tackles two problems: It reduces the danger of overfitting, because it reduces the possibilities of combining arbitrary features that may occur otherwise at any time in the recorded history and, secondly, it provides the necessary background information for the user when inspecting the result.

The remainder of the paper is organized as follows: In Sect. 2 we briefly discuss the representation of temporal data and review related work. The classifier we are going to use in this paper is reviewed in Sect. 3, while an approach to provide the aforementioned background information is discussed in Sect. 4. Results on the libras data set are discussed in Sect. 5. Section 6 finally concludes the paper.

2 Representation and Related Work

Rather than considering values individually, we employ temporal abstractions such as 'rising temperature', 'connection established', 'low user activity', or 'increased variance'. We thus describe the evolution by means of temporal predicates: denoting the temporal dimension by $\mathbb{T}$, a temporal predicate P_l is a function $P_l : \mathbb{T} \to \mathbb{B}$, where l is called the label of the predicate P. Examples for predicates (and especially their label) were given above. The choice of predicates is domain dependent and part of the feature selection step in data mining. A set of predicates (which we will call history H) may be depicted by plotting them against the temporal dimension (cf. Fig. 1). The use and visualization of *temporal abstractions* has a long tradition in the medical domain [9]. Note that in contrast to stream mining approaches, where a single but potentially infinite stream of data is considered, we assume that multiple (finite) labelled histories are available.

Various ways of defining patterns in a stream of labeled intervals have been proposed in the literature, many of them relying on Allen's interval relationships [1] (cf. Fig. 2) or variants thereof. Some approaches define a history by specifying the exact relationship for every pair of intervals [3], others allow for a set of possible relationships [4]. The representation by sequences of chords [6] uses a partially ordered sequence of simultaneous (sub)intervals to define a pattern. Other proposals consider a different set of interval relationships or specify the relationship between temporal intervals only partially [5].

While these approaches have their individual strengths, they also have their weaknesses even when it comes to represent simple situations. Thinking of predicting a certain state of some network server (breakdown, overload, malfunction, etc.) on the

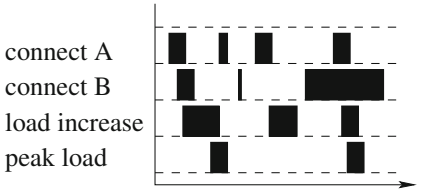


Fig. 1 Representation of Evolving Data: the black rectangles denote the intervals when the predicate holds (labels on the left)

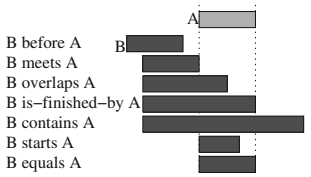


Fig. 2 Thirteen possible relationships between two intervals. The inverse relationships (before $\leftrightarrow$ after) have been omitted.

history of, say, the last 24 hours, a situation as simple as "there was only one connection to server A" (during the last n hours) is usually prohibitive to discover using approaches based on association rules [3, 6], as they count occurrences of events and often rely on a quickly decreasing count of co-occurrences, which forbids an inclusion of *absent features* during counting. A situation like "at some point in time, both A and B hold" is a challenge to approaches such as [3], because they rely on explicitly specified interval relationships (which are ambiguous in this case). Temporal constraints "the connection to A was lost for at least 4 hours" or "... at most 4 hours" are usually ignored or introduced in a postprocessing step.

3 Representing and Classifying Temporal Data

To support an intuitive understanding we choose a rule-based approach where the conclusion part predicts the class and the premise of the rule contains a pattern that has to be matched to a given history. In [7] a notion of a pattern, called *template history*, has been introduced. A template history may be visualized as in Fig. 1, but this time a black box is understood as a constraint that has to be fulfilled by a matching history. The constraints on the presence of temporal abstractions are not fixed in time to compensate dilational and translational effects, only the order in which the constraints have to be fulfilled must be preserved. A template is thus decomposed into a number of n successive blocks whose absolute duration may vary from case to case. Together with a selection of m temporal predicates, we obtain an $m \times n$ matrix C where each cell $C_{i,j}$ represents a constraint on the i^{th} predicate in the j^{th} block (cf. Fig. 3). We distinguish between four different constraints:

Definition 1 (predicate constraint). Given a temporal interval $T \subseteq \mathbb{T}$ and a predicate P, we say (a) *P is present during T* if $\forall t \in T : P(t)$, (b) *P is absent during T* if $\forall t \in T : \neg P(t)$, (c) *$P$ exists during T* if $\exists t \in T : P(t)$ and (d) *P disappears during T* if $\exists t \in T : \neg P(t)$. If no condition is posed, we say *P is unconstrained during T*. By $\mathbb{C}$ we denote the set of constraints $\{$present, absent, exists, disappears, unconstrained$\}$.

Besides the constraints in the cells of the matrix, we may additionally constrain the duration of each block:

Definition 2 (template). A tuple $T = (L,n,C,D)$ is called a template if L is a set of labels, $n \in \mathbb{N}$, $C : L \times \{1,..,n\} \to \mathbb{C}$ and $D : \{1,..,n\} \to (\mathbb{T} \cup \{\infty\})^2$ satisfying $1 \leq d_{\min} \leq d_{\max}$ for any $D(i) = (d_{\min}, d_{\max})$, $1 \leq i \leq n$. The map C constrains the predicate in each block, the map D contrains the block duration.

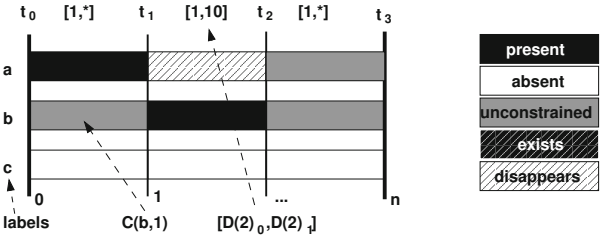


Fig. 3 Illustration of the template definition. The predicate constraints are color-coded.

Figure 3 shows an example template with $n = 3$ blocks, defined by four time points (vertical lines), where the leftmost and rightmost time point shall always represent the start and end of the history. The bottom row declares that a predicate P_c is absent in the whole history. Somewhere in the history (2nd block), P_b is present (P_b may be present or not elsewhere (=unconstrained)). P_a is present from the very beginning, but disappears while P_c is present in the 2nd block. The duration of the first block is arbitrary, the second block takes 1 to 10 time units ($D(2) = (1,10)$), the last block may again have any (positive) duration ('*' represents '∞').

Matching a template to a history involves the determination of points t_i in time (temporal alignment) such that all constraints hold.

Definition 3 (match). Let $T = (L,n,C,D)$ be a template and H be a history. Let $[t_{\min}, t_{\max}]$ be the smallest interval subsuming $\cup_{P \in H} dom(P)$. T matches a history H if and only if (a) there is a predicate $P_l \in H$ for every $l \in L$, (b) there are $t_i \in \mathbb{T}$, $0 \leq i \leq n$, with $t_0 = t_{\min}$, $t_i \leq t_{i+1}$, $t_n = t_{\max}$, (c) for every $l \in L$ and $i \in \{1,..,n\}$ the constraint $C(l,i)$ holds for P_l within $[t_i, t_{i+1})$ and finally (d) for all $1 \leq i \leq n$: $t_i - t_{i-1} \in [d_{\min}, d_{\max}]$ with $(d_{\min}, d_{\max}) = D(i)$.

In [7] we proposed a method to explore the space of templates to find good discriminators between differently labeled histories. The search algorithm implements a general-to-specific search: It begins with an empty pattern and specializes it further to improve some measure of interestingness (we used the J-measure [10] as it balances the generality (applicability of the rule) and the interestingness (deviation from a priori knowledge)). The initial template that matches all histories consists of one block, all predicate constraints are *unconstrained* and so are the temporal constraints $(1,\infty)$. While a propositional rule can only be specialized by an additional condition (like *outlook=sunny*), there are at least three ways to specialize a template: we may look at it in a finer resolution (by subdividing the temporal axis further), we may change or add a predicate constraint (for some label and block), or

may introduce or change an existing temporal constraint. We thus have chosen three different specialization operators to address each of these aspects. The general idea for all refinement operators is to search for specializations that improve the measure of interestingness. A more detailed description of the operators and the quality of the learned patterns can be found in [7].

4 Providing Background Information

The idea of providing 'temporal context' in a template is to find some (most specific) pattern that matches all instances of a given class. Such a pattern may be used as a starting point for the beam search mentioned in Sect. 3. The problem of finding such a pattern is closely related to the alignment of multiple sequences, which is known to be NP-complete [11]. As all instances of the same class may in principle share a considerable number of blocks, the use of pattern mining algorithms that enumerate subpatterns is prohibitive because the number of subpatterns grows exponentially with the length of the sequence. However, we do not rely on the *optimal* or even the *most specific* pattern, but assume that any pattern that is shared by all instances of the same class will help to provide contextual background. Therefore we are duly satisfied with an approximate or heuristic solution to this problem. One possible approach will be discussed in the remainder of this section, but we do not claim any specific properties or advantages of this solutions: but our intention is to demonstrate that (any) common subpattern is potentially useful.

The idea behind our simple heuristic method is to exploit the fact that each instance itself should match the sought common subpattern – and that we thus may identify it by simplifying the instance subsequently. At first, an arbitrary selected history is transformed into a template history: Whenever a predicate changes its value, we introduce a new block. If the predicate holds during the block, we place a *present* constraint in the respective block, otherwise we leave it *unconstrained*. A copy of this template history is created where all predicate constraints are set to *unconstrained*, which is trivially matched by all instances. Next, we transfer the *present* constraints (one by one) from the instance pattern to the (initially blank) copy and only keep it if it still matches all instances of its class.

For example: Given the three different histories shown in Fig. 4, we want to find a common subpattern, shared by all three histories. We start by using (a) as the start

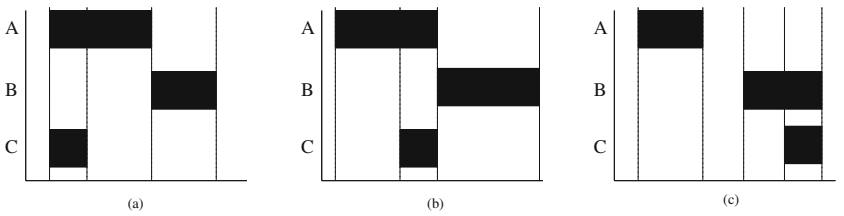


Fig. 4 Three sample histories for the starting pattern algorithm

instance (please note that every instance could be picked). In the first step we create the template history by counting the blocks (segments between the dashed lines because there is at least one predicate change) and convert it into the history shown in Fig. 5. Finally we change all predicate constraints to *unconstrained* as shown in Fig. 6.

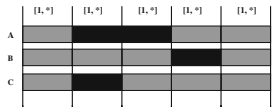


Fig. 5 Sequence (Fig. 4(a)) transformed into a template history

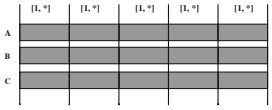


Fig. 6 Template history (Fig. 5) with all predicate constraints changed to *unconstrained*

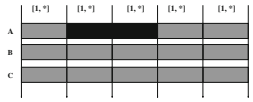


Fig. 7 Starting pattern after adding the A-Label intervals to the starting pattern in Fig. 6

In the second step we add constraints to the pattern and check if the resulting pattern still matches all instances. Therefore we go through the original template history row by row and transfer the *present*-constraints to the pattern. Furthermore we add a new *unconstrained* block before and/or after the modified block to relax the required predicate relationships. We obtain four possible patterns in total – if more than one turns out to match all histories we choose the most specific one. For label A the algorithm adds the *present* constraints as shown in Fig. 7 directly to the pattern, so we inspect the refinements for label B in more detail. The four possible patterns shown in Fig. 8 are created as the possible refinements. Evaluating the first pattern shows that the instance in Fig. 4(c) is not matched anymore because the relation '*A meets C*' is not present (but '*A before B*'). The second pattern matches all sequences because the meet-relationship has been relaxed by the intermediate *unconstrained*-block. The remaining two patterns duplicate the final *unconstrained*-block but do not add any substantial differences. A further refinement is not possible, as there is no position for *B* that matches all three histories in Fig. 4.

Drawbacks of heuristic approach. From the example above we also recognize the drawback of this approach. The resulting patterns depend on the order in which the labels are added to the pattern. If we had started with adding a *present* constraint for predicate *C* we would not be able to add any more constraints, because all other intervals occur in different relationships to *C*.

5 Experimental Evaluation

We applied our algorithm to the libras movement data set from the UCI repository [2]. It contains 15 different signs described by their characteristic hand movements over 45 time frames, where the current x- and y-positions of the hand were recorded. There are 24 instances per sign, 360 in total.

Data preperation & evaluation settings. In a first step we have manually inspected all hand movements and removed clear outliers and incomplete movements, such

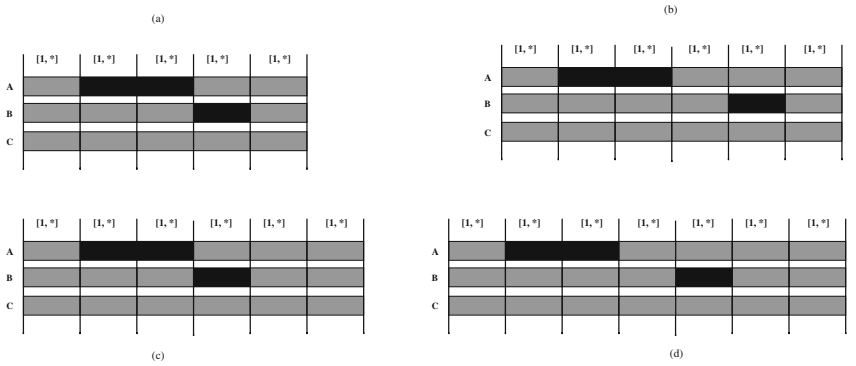


Fig. 8 Four template histories tested by the algorithm by adding B to the pattern shown in Fig. 7

that not only parts of the hand movement appear in each class but the complete sign is visible. We have subsequently extracted predicates that represent the hand movement, e.g., the speed of the movement (overall speed and separate movement in x- and y-direction). We used a priori defined thresholds and the following labels only:

- x-movement: fast left ($--$), left ($-$), constant (o), right (+), fast right (++).
- y-movement: fast down ($--$), down ($-$), constant (o), up (+), fast up (++)
- x/y-moveall: fast ($++$), normal ($-$) (this label is the same as x/y-movement without distinguishing between left/right resp. down/up).
- Curve: nearly same direction (o), middle change of direction ($+$), abrupt change of direction($++$)

For example, a fast hand movement to the upper left may be recognized by observing predicates x-movement $--$ and y-movement $++$ at the same time.

We divided the preprocessed data into training (66%) and test (33%) data. For each sign we constructed a shared pattern and refined it using the classifier in [7] on the training set. The signs #10 and #12 were merged to just one class #10, because the set of features we had chosen was not suited to distinguish between these two hand movements. For the evaluation against the test set, we matched an instance against all obtained patterns – if an instance matches only a single rule pattern, the classification rule predicts the class; if no unique pattern matches, we classify it as "cannot predict".

5.1 Effect on Interpretability

Before discussing the classification performance we start by comparing the learned template histories for three different hand movements shown in Fig. 9, 10 and 11. We want to demonstrate the usefulness of the identified 'common pattern' for aligning it with the user's mental model of the considered process.

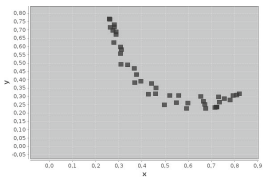


Fig. 9 Example hand movement for sign #1

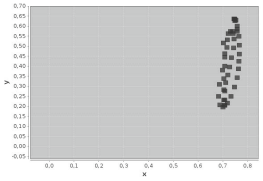


Fig. 10 Example hand movement for sign #3

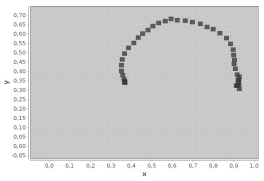


Fig. 11 Example hand movement for sign #14

Sign #1. The pattern for sign #1 found without a start pattern is shown in Fig. 12. By inspection of this pattern, it is quite difficult to recover the actual hand movement, because it consists of many *absent*-constraints, which are difficult to align with a mental model of the hand movement. Furthermore, the first and last blocks consist of *unconstrained*-conditions only. Thus we do not get any information about what may happen *before* or *after* the pattern or at what time it might occur.

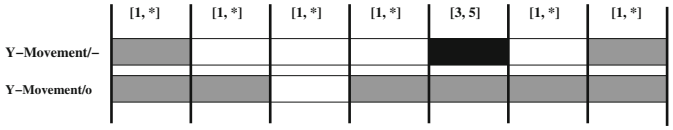


Fig. 12 Pattern found by the beam search without a starting pattern for sign #1

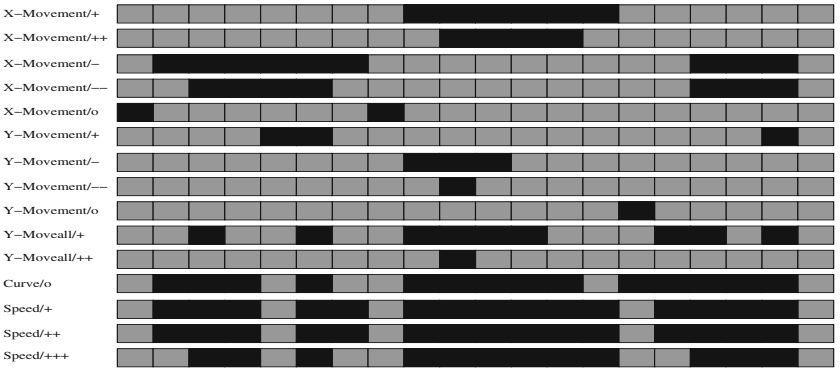


Fig. 13 Pattern found by the beam search with the help of a starting pattern for sign #1

Fig. 13 shows the pattern found with the help of a starting pattern: the many *present*-constraints support the user in understanding the actual hand movement. The pattern describes the hand movement almost completely: a fast left-right-left movement (*present* x-movement/–, x-movement/++ and x-movement/–) combined with an up-down-up movement (*present* y-movement/+, y-movement/- and

x-movement/+). We also observe periods of 'high speed movements' and low speed when changing directions. In total the key features of the sign #1 as shown in Fig. 9 are well reflected.

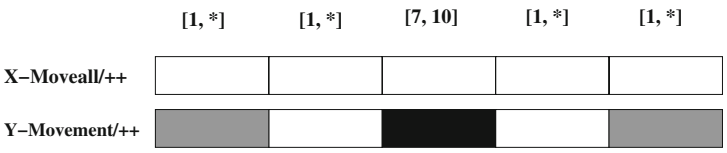


Fig. 14 Pattern found by the beam search without a starting pattern for sign #3

Sign #3. Fig. 14 shows the pattern learned for the sign #3 without a starting pattern. It is again a simple pattern which forbids the occurrence of a fast left or right movement (*absent x-moveall/++*) at any time and requires a fast upwards movement for 7-10 time frames in the middle of the sign (*absent* to *present* to *absent* constraint for y-movement/++). Again, this pattern does a good job in discriminating sign #3 from all other signs, but it does not help the user to get an impression of sign #3, because it mainly carries information about which predicates are *not allowed* rather than which are *required*.

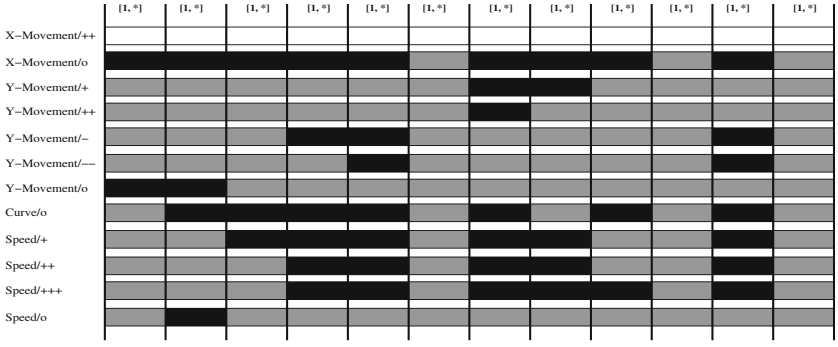


Fig. 15 Pattern found by the beam search with a starting pattern for sign #3

The pattern which was learned with the help of a starting pattern (Fig. 15) reveals the hand movement pretty well. We recognize that the pattern falls into three parts with *unconstrained* blocks (6th and 10th block), which allow for gaps between the three parts. In the first part the pattern requires no noticeable movement (*present* X- and y-movement/o) at the beginning, and is followed by a downward move (y-movement $--$), an upward move in the second part, and a downward move in the third part again. No movements to the left or right are allowed as the 'x-movement/o' predicate is *present* most of the time. The *absent* constraint for 'x-movement/++'

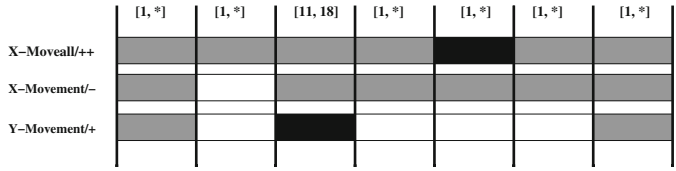


Fig. 16 Pattern found by the beam search without a starting pattern for sign #14

(top row) was added during the beam search refinement to better discriminate the pattern from all other classes.

Sign #14. Finally we inspect the results for sign #14 (cf. Fig. 11). The pattern obtained without using a starting pattern is shown in Fig. 16 and describes an upward move of 11 to 18 time frames and no such upward move before or afterwards. The downward move (which occurs later) is not part of this pattern, because it did not help to discriminate sign #14 from other signs, but it would definitely help the user to interpret the pattern and associate it with Fig. 11.

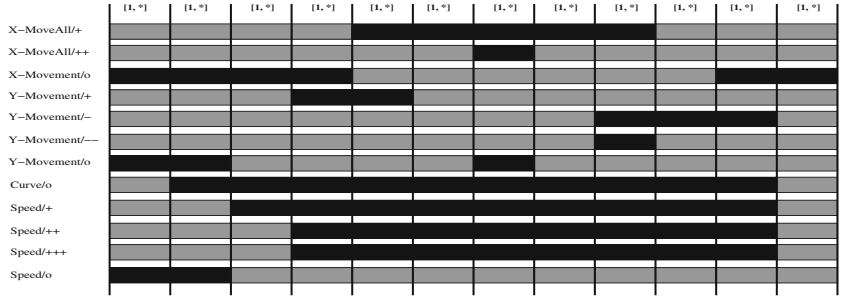


Fig. 17 Pattern found by the beam search with a starting pattern for sign #14

In Fig. 17 we see the pattern found when using the starting pattern. The pattern is *fully connected*, there is no block with only *unconstrained* predicates. Thus the pattern describes the whole movement without any gaps, which is particularly helpful in reconstructing the hand movement. The pattern requires that there is no movement at the beginning (x/y-movement/o *present*). During blocks 3 and 4 the template describes an upward move (*present* y-movement/+) followed by a combined upward move to the right or left (*present* x-moveall/+ and y-movement/+). In block seven the upward move stops because y-movement/o has to be *present* and the movement in x-direction accelerates (as *present* x-moveall/++ appears). During blocks 8-10, the hand movement in the x-direction decelerates and starts to move downwards. In the last two blocks the x-movement disappears (as x-movement/o *present* appears) and the downwards move gets slower as well. If we now take into account that during the whole hand movement the speed of the hand is very high and there are no abrupt change of directions because *curve/o* holds, we are able to

interpret the pattern as a half circle movement. Another interesting aspect is that the pattern does not state a concrete direction of the x-movement. This is due to the fact that the sign could be drawn from right to left or from left to right.

Some hand movements appear easier to understand by inspecting the plots (e.g. Fig. 11) rather than the obtained patterns (e.g. Fig. 17). But this is only true because the underlying predicates have been extracted from two-dimensional hand movements. In general the data source may consist of more dimensions, mixed binary and numerical sensors, etc., such that no *condensed* representation as in Fig. 11 is possible. We have chosen the libras data set to illustrate that the proposed history templates actually help the user to grasp *what is going on in the data.*

5.2 *Effect on Classification Performance*

Having discussed the effect on the interpretability, we we now investigate the effect on the classification results. The confusion matrices are shown in Fig. 18 (without starting patterns) and Fig. 19 (with starting patterns).

class	1	2	3	4	5	6	7	8	9	10	11	13	14	15	cannot predict
1	3														1
2		5													1
3			4												
4				3											2
5					2										3
6						1									4
7	1					4	1								
8							3								
9								8							
10									10						3
11											5				3
13												4			4
14													5		
15														5	

Fig. 18 Confusion matrix for the learned patterns without starting pattern with accuracy: 71.765% and error-rate: 28.235%

class	1	2	3	4	5	6	7	8	9	10	11	13	14	15	cannot predict
1	4														
2		6													
3			4												
4				5											
5					5										
6						4									1
7							4								2
8								3							
9									8						
10										11					2
11											7				1
13												8			
14													5		
15														5	

Fig. 19 Confusion matrix for the learned patterns with starting pattern with accuracy: 92.941% and error-rate: 7.059%

We can see that accuracy improves by around 21 percent. One reason is the greedy nature of the beam search. During the beam search only constraints that increase the J-measure are added to the pattern, thus refinements which require multiple steps to increase the measure are not found easily due to the myope of the search algorithm. By providing the starting pattern, it is more likely that a critical constraint can be placed right where it is needed, because the basic outline of the pattern is already present right from the beginning. As the intial pattern is constructed such that it matches all histories of one class, the danger of overfitting is not increased despite the high complexity of the pattern.

6 Conclusion

We have investigated the problem of deriving classification rules for temporal or sequential data. The employed classifier operates by successively refining a given

pattern to better distinguish between the classes. Instead of learning the patterns for each class *from scratch*, we propose to derive a starting pattern, which consists of those parts that are shared among all instances of the same class (a representative for this class, which has not necessarily any discriminative power). The experimental evaluation has shown that this step not only improves the interpretability of the obtained patterns, but also improves the classification results. Increasing the explanatory power of the patterns [8] and reducing the complexity of searching the starting pattern are topics to be addressed in future work.

References

[1] Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM 26(11), 832–843 (1983)
[2] Frank, A., Asuncion, A.: UCI machine learning repository. University of California, Irvine (2010)
[3] Höppner, F., Klawonn, F.: Finding informative rules in interval sequences. Intelligent Data Analysis – An International Journal 6(3), 237–256 (2002)
[4] Höppner, F., Topp, A.: Classification Based on the Trace of Variables over Time. In: Yin, H., Tino, P., Corchado, E., Byrne, W., Yao, X. (eds.) IDEAL 2007. LNCS, vol. 4881, pp. 739–749. Springer, Heidelberg (2007)
[5] Kam, P.S., Fu, A.W.C.: Discovering Temporal Patterns for Interval-Based Events. In: Kambayashi, Y., Mohania, M., Tjoa, A.M. (eds.) DaWaK 2000. LNCS, vol. 1874, pp. 317–326. Springer, Heidelberg (2000)
[6] Mörchen, F.: Time series knowledge mining. Ph.D. thesis, Philipps University Marburg (2006)
[7] Peter, S., Höppner, F.: Finding Temporal Patterns Using Constraints on (Partial) Absence, Presence and Duration. In: Setchi, R., Jordanov, I., Howlett, R.J., Jain, L.C. (eds.) KES 2010. LNCS, vol. 6276, pp. 442–451. Springer, Heidelberg (2010)
[8] Peter, S., Höppner, F., Berthold, M.R.: Pattern graphs: A knowledge-based tool for multivariate temporal pattern retrieval. In: Proc. IEEE Conf. Intelligent Systems. IEEE (2012)
[9] Shahar, Y., Musen, M.A.: RÉSUMÉ: A temporal abstraction system for patient monitoring. Computers and Biomedical Research 26, 155–273 (1993)
[10] Smyth, P., Goodman, R.M.: An information theoretic approach to rule induction from databases. IEEE Trans. Knowledge Discovery and Engineering 4(4), 301–316 (1992)
[11] Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. Journal of Computational Biology 1(4), 337–348 (1994)

Comprehensiveness of Linguistic Data Summaries: A Crucial Role of Protoforms

Janusz Kacprzyk and Sławomir Zadrożny

Abstract. We show first the essence of our approach to linguistic database summaries, equated with linguistically quantified propositions in Zadeh's sense and mined through the use of a fuzzy querying interface to a database. We recast the problem from the perspective of comprehensiveness of patterns derived by linguistic data summaries. Motivated by Michalski's [21] seminal approach to the comprehensiveness of data mining and machine learning results in which he advocates the use of natural language, we advocate the use of linguistic summaries which provide a new quality and an exceptional human consistency and comprehensiveness. We illustrate our analysis by two examples related to the linguistic summarization of both static and dynamic data in the area of analysis of innovativeness of companies and of Web server log files.

1 Introduction

The purpose of this paper is to briefly touch upon an important and interesting issue of *comprehensiveness* (or comprehensibility) of data mining, or – more generally – data analysis and even knowledge discovery. The perspective adopted will however be very specific, much less general than, for instance, some other works on the comprehensibility of data mining exemplified by Zhou [32], Pryke and Beale [22], Fish, Gruber and Sick [4], etc.

While speaking about quality criteria for data mining or knowledge discovery tools and techniques, we usually mention that they should provide: (1) novelty, (2) correctness, (3) generality, (4) usefulness, and (5) comprehensiveness; clearly, some authors add some other criteria.

Janusz Kacprzyk · Sławomir Zadrożny
Systems Research Institute, Polish Academy of Sciences, 01-447 Warsaw, Poland
e-mail: {kacprzyk,zadroznykacprzyk}@ibspan.waw.pl

Sławomir Zadrożny
Warsaw School of Information Technology, 01-447 Warszawa, Poland
e-mail: zadrozny@wit.edu.pl

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 207–221.
springerlink.com

In this paper we will be concerned about some aspects of *comprehensiveness* which is basically meant as that for the user, who is almost always not a data mining expert, the patterns produced as a result of data mining should be clear and understandable. To be more specific, with any data mining algorithm we usually associate, explicitly or implicitly, some more or less formal representations for the patterns it is meant to produce (mine). It is therefore clear that to be usable and implementable, the data mining algorithm should produce patterns whose representations are comprehensible, that is, the algorithm should encode the patterns derived in a form understandable to the human user.

The problem of comprehensiveness of data analysis, data mining, machine learning, etc. results (patterns) had been known for some time, and it had been presumably Michalski who already in 1982 devised the so called *postulate of comprehensibility* whose essence can be summarized as (cf. Michalski [21]): "...The results of computer induction should be symbolic descriptions of given entities, semantically and structurally similar to those a human expert might produce observing the same entities. Components of these descriptions should be comprehensible as single "chunks" of information, directly interpretable in natural language, and should relate quantitative and qualitative concepts in an integrated fashion ...".

Michalski's statement, and – more generally – his vision, has had a great impact on machine learning, data mining, etc. research, and has also played an extremely important role by triggering our research summarized in this short paper as it will be discussed later.

Later, many people have further extended Michalski's idea of comprehensiveness, and just to give some more relevant example, we can cite Craven and Shavlik [3] who stated as the main reasons for the importance of comprehensiveness of machine learning algorithms: (1) To be confident in the performance and usefulness of the algorithms, and hence to be willing to use them, the users have to understand how the result is obtained and what it says, (2) By assumption, the results to be produced by a data mining (machine learning, etc.) algorithm should be novel and unexpected, in one sense or another, and these results can only be accessible to the human if they are understandable, (3) It is usually assumed that the mining patterns may imply some action to be taken, and in such a case their comprehensiveness is clearly crucial, (4) The patterns mined may provide much insight into a possibility of devising a better feature representation, and their comprehensiveness is again crucial, (5) Data mining algorithms can be employed for refining knowledge (theories) about a domain or field in question, and it is crucial to be able to express changes indicated as a result of data mining.

Therefore, it is obvious that data mining tools and techniques of a good comprehensibility are extremely desirable, if not crucial. Unfortunately, most of them are not very comprehensible *per se* and some additional mechanisms, sometimes *trickeries*, should be used to enhance their comprehensibility.

In this paper we will follow the above line of reasoning but assume some particular perspective which has been advocated and indicated in the excerpt of Michalski's paper already mentioned above which will be repeated below for convenience: "...The results of computer induction should be symbolic descriptions of given

entities, semantically and structurally similar to those a human expert might produce observing the same entities. Components of these descriptions should be comprehensible as single "chunks" of information, directly interpretable in natural language, and should relate quantitative and qualitative concepts in an integrated fashion . . . ". It can be readily seen that he makes a direct relation to natural language the use of which he indicates as a prerequisite for the comprehensibility. This is in line with our philosophy which is based on the obvious fact that natural language is the only fully natural way of articulation and communication of the human being and should therefore be employed, explicitly or implicitly, in data mining tools and techniques, requirements' specifications, and finally patterns (results) obtained.

In our particular case, this boils down to the use of *linguistic data summaries* which for years have been our field of interest, both in the sense of theory and real world applications. They are meant to summarize the very meaning of a (usually huge) set of (numeric, in our case) data via a simple and short statement(s) in (quasi)natural language, exemplified by "most young and highly qualified people earn salaries" in the case of a personnel database. It is clear that the need for summarization is due to abundance of data that is beyond human cognition and comprehension, and that for a human being the only fully natural means of communication is natural language. We will consider the linguistic data(base) summaries introduced by Yager [24], then advanced by Kacprzyk and Yager [7], and Kacprzyk, Yager and Zadrożny [18], and implemented in Kacprzyk and Zadrożny [10, 13, 14, 15]. They are assumed to be linguistically quantified propositions.

Even if conceptually the linguistic data summaries are simple and intuitively appealing (perfectly comprehensible!), their derivation (mining) is difficult due to very many possible forms, linguistic terms and expressions, etc. And, *a fortiori*, an automatic expression of the real human interest and intention with respect to a linguistic summary is questionable. We adopt our general approach (cf. Kacprzyk and Zadrożny [10, 12]) of an interactive approach via the use of our FQUERY for Access, a fuzzy querying add-on (see Kacprzyk and Zadrożny's [9, 8, 11] and also Zadrożny *et al.* [31]).

In this respect, we show that by relating various types of linguistic summaries to fuzzy queries, with various known and sought elements, we end up with a hierarchy of Zadeh's [28] protoforms of linguistic data summaries. We discuss the power of protoforms, and indicate ways of an automatic generation of linguistic summaries for various protoforms. We mention a possible use of association rule mining. We will discuss this but now from the novel point of view of comprehensiveness. Moreover, we will give some examples of comprehensible protoforms to users in various domains.

2 Linguistic Data Summaries: An Approach Based on Fuzzy Logic with Linguistic Quantifiers

In our works we have been using the basic Yager's [24] approach, through its constructive form by Kacprzyk and Yager [7], and Kacprzyk, Yager and Zadrożny [18], and implemented in Kacprzyk and Zadrożny [10, 11, 14], in which we have: (1) V,

a quality (attribute) of interest, e.g., salary in a database of workers, (2) a set of objects (records) y_i that manifest quality V, e.g., the set of workers; hence $V(y_i)$ are values of quality V for objects y_i, and (3) $Y = \{V(y_1), \ldots, V(y_m)\}$ is a set of m pieces of data (the "database" in question).

A linguistic summary of a data set consists of:

- a *summarizer* S, i.e., a fuzzy predicate describing a property, simple or compound, of the objects that may be of interest to the user and is possibly shared by a reasonable quantity (cf. description of Q below) of objects (e.g., "young", extendable to "young and well paid", etc.),
- a *qualifier* K, i.e., another fuzzy predicate describing a range of objects to which the summarizer applies (e.g., "young", extendable to "young and well paid", etc.); effectively it turns the set Y which a summary is to summarize into its fuzzy subset K (e.g., a summary thanks to a qualifier "young" may describe a property (expressed by the summarizer) shared by, e.g., a majority of "young employees" instead of the employees in general; if $K = Y$ (identifying a predicate with its extension) then the qualifier is omitted in the specification of the linguistic summary,
- a *quantity in agreement* Q given as a fuzzy linguistic quantifier (e.g., *most*), which expresses how many objects from among those satisfying a qualifier K share a property expressed by a summarizer S,
- *truth degree* T — e.g., 0.7, meant as a truth of a linguistically quantified proposition $Q_{y \in Y}(K(y), S(y))$ as, e.g., "T(*most young employees are well-paid*) $= 0.7$".

The truth degree is equated with the truth value (from $[0,1]$) of a linguistically quantified statement which may be done by using Zadeh's [27] calculus of linguistically quantified propositions (cf. Zadeh and Kacprzyk [29]), and this will be used here, too; cf. also Yager's [25] OWA operators (cf. also Yager and Kacprzyk [26]).

By using Zadeh's calculus of linguistically quantified propositions we can calculate the truth value of the propositions:

$$Q_{y \in Y}\, S(y) \quad \text{(e.g., "Most elements of } Y \text{ possess property } S\text{")} \tag{1}$$

or, more generally,

$$Q_{y \in Y}\, (K(y), S(y)) \quad \text{(e.g., "Most elements of } Y \text{ with property } K \text{ possess also property } S\text{")} \tag{2}$$

using the following formulas, respectively:

$$\text{truth}(Q\, S(y)) = \mu_Q\left(\frac{\sum \text{Count}(S)}{\sum \text{Count}(Y)}\right) = \mu_Q\left(\frac{1}{m}\sum_{i=1}^{m} \mu_S(y_i)\right) \tag{3}$$

$$\begin{aligned} &\text{truth}(Q\,(K(y), S(y))) = \\ &= \mu_Q\left(\frac{\sum \text{Count}(S \cap K)}{\sum \text{Count}(K)}\right) = \mu_Q\left(\frac{\sum_{i=1}^{m} (\mu_S(y_i) \wedge \mu_k(y_i))}{\sum_{i=1}^{m} \mu_K(y_i)}\right) \end{aligned} \tag{4}$$

where $m = \text{card}(Y)$, $\sum \text{Count}(A) = \sum_{y_i \in Y} \mu_A(y_i)$, $\sum_{i=1}^{m} \mu_k(y_i) \neq 0$, and $\wedge$ is a t-norm.

The basic validity criterion, i.e., the truth degree T, given by (3) or (4) is the most important and widely employed. One of the reasons is its high comprehensiveness because virtually all users, even novice, can comprehend what truth means, and the fact that a linguistic summary to be acceptable and meaningful should have a high truth degree. It is, however, easy to see that the truth degree alone is too weak to be a criterion for the goodness of a linguistic summary, and hence some other quality (validity) criteria have been proposed by, e.g., Yager's [24] measure of informativeness, and then five additional measures proposed by Kacprzyk and Yager [7] and Kacprzyk, Yager and Zadrożny [18]: truth, degrees of imprecision, covering and appropriateness, and a length of a summary. For even more measures, see Kacprzyk, Wilbik and Zadrożny [19, 20]. Unfortunately, though all those measures do capture very well, much better than the truth degree alone, how good a linguistic summary is, the comprehensiveness of some of them to an average user may be questionable. Among them, the length of a summary which basically boils down to the complexity of the summarizer and qualifier, is surely closely positively correlated with the intuitively understood comprehensibility.

The very advantage of the linguistic summaries with respect to their comprehensibility is, as advocated earlier, their use of the linguistic terms. It is especially true if the linguistic terms used to compose summaries have a clear meaning to the user. This may be achieved when a dictionary of such terms is used by the user also for some other purposes, securing the clarification of their semantics, and what is even more important, allowing for a convenient tuning of their meaning in the framework of the representation assumed (here: using fuzzy logic). In particular, such a synergestic effect may be obtained when a linguistic summaries mining tool is combined with a fuzzy flexible querying interface what will be exemplified in what follows.

The real problem is clearly how to generate the best summary (or summaries). An exhaustive search can obviously be computationally prohibitive, and some implicit enumeration type schemes should be used. We will discuss this in some detail in the next section.

3 Mining Linguistic Data Summaries through Fuzzy Querying: A Protoform Based Analysis

Obviously, it is very difficult to automatically detect what (in the sense of a linguistic summary) is interesting, intended, useful, etc. to the user. In Kacprzyk and Zadrożny [12] we proposed a natural solution, that is, an interactive approach for the definition of elements of an intended linguistic summary via a user interface of a fuzzy querying add-on. The roots are our previous papers on the use of fuzzy logic in querying numerical databases (cf. Kacprzyk and Ziółkowski [16], Kacprzyk, Zadrożny and Ziółkowski [17]) by using imprecisely specified requests which led to our FQUERY for Access, an add-in to Microsoft Access® that makes it possible to use fuzzy linguistic terms in database queries such as *numerical fuzzy values*, exemplified by

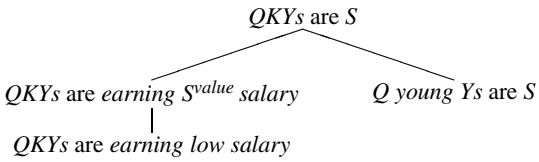


Fig. 1 An example of a part of a hierarchy of protoforms

low in "profitability is *low*", *fuzzy relations*, exemplified by *much greater than* in "income is *much greater than* spending", and *linguistic quantifiers*, exemplified by *most* in "*most* conditions have to be met". These fuzzy linguistic terms are building blocks of fuzzy queries and are represented as fuzzy sets.

In the context of linguistic summaries equated with linguistically quantified propositions, of a particular importance are *linguistic quantifiers* which provide for a more flexible aggregation of simple conditions in fuzzy queries, exemplified by, instead of requiring that all simple conditions are met, one may indicate that *most* of them are to be met.

The definition, processing and storage of the above mentioned linguistic terms was implemented in our FQUERY for Access package (cf. Kacprzyk and Zadrożny [9, 8, 11]).

It is easy to see that fuzzy queries (with linguistic quantifiers) directly correspond to linguistic summaries in the sense assumed here. Thus, a linguistic summary may be derived as follows: (1) the user formulates a set of linguistic summaries of interest (relevance) using the fuzzy querying add-on, (2) the system retrieves records from the database and calculates the validity of each summary in question, and (3) a most appropriate linguistic summary is chosen.

Obviously, to make this derivation process effective and efficient, operationally, some standardized forms of linguistic summaries would be desirable, and this is provided by Zadeh's *protoform* viewed as an abstract prototype of a linguistic summary given by Eq. (1) or Eq. (2).

Technically, for the generation of linguistic summaries it is convenient to consider the summarizer (and the qualifier) as an abstract fuzzy logic statement "X IS A", where X is a placeholder for an attribute of objects in Y and A is a placeholder for a fuzzy set (linguistic term) determining its value as, e.g., "age IS young" or "salary IS A". The former summarizer is fully instantiated, while the latter still has an abstract attribute value (A).

The protoforms (in Zadeh's sense) may obviously form a hierarchy, and hence we can define lower level (less abstract) protoforms, for instance replacing Q by a specific linguistic quantifier, "most", and we get: "Most Ys are S" for (1) and "Most KYs are S" for (2). Zadeh's protoforms may conveniently be used as a fundamental element of the user interface in that the user selects a protoform of a linguistic summary from that hierarchy and then the system instantiates the selected protoform in all possible ways, replacing abstract symbols by chosen fuzzy values and linguistic quantifiers stored in a dictionary. A part of such a hierarchy of protoforms is shown in Figure 1. At the top we have a completely abstract protoform; in a

Table 1 A taxonomy of linguistic summaries

Type	Given	Sought	Remarks
1	S	Q	Simple summaries through ad-hoc queries
2	$S\ K$	Q	Conditional summaries through ad-hoc queries
3	$Q\ S^{structure}$	S^{value}	Simple value oriented summaries
4	$Q\ S^{structure}\ B$	S^{value}	Conditional value oriented summaries
5	Nothing	$S\ K\ Q$	General fuzzy rules

protoform to the right, the qualifier K is instantiated to "age IS young"; in the one to the left, summarizer S is first instantiated to "salary IS S^{value}", i.e., the attribute of the summarizer is selected to be "salary" but its value is not determined; then this protoform is further instantiated to fully specify the summarizer using "low" as the value of "salary". Notice that by relating the linguistic summaries to the protoforms we maintain a high degree of comprehensiveness because we operate within the same structure of the protoform (linguistic summary) and just instantiate or generalize a particular element of the summary. The user is therefore not forced to leave his/her "safe" area of expertise, in the sense of a proper type of protoform of a linguistic summary that is comprehensible in a particular domain; this will be clearer later on when we will quote some examples of what may a proper protoform of a linguistic summary in some chosen domains.

Therefore, the more abstract forms of protoforms correspond to cases in which we assume less about the summaries to be mined. At the one extreme, we (1) assume a totally abstract (top) protoform, or (2) assume that all elements of a protoform are given, i.e., all attributes and all linguistic terms expressing their values are fixed. In the former case data summarization by a "brute force" full search would be extremely time-consuming, but might produce interesting, unexpected patterns, and in the latter case the user guesses in fact a good candidate summary but the evaluation is simple, related to ad hoc queries.

This classification is shown in Table 1 in which 5 basic types of linguistic summaries are shown, corresponding to protoforms of a more and more abstract form; $S^{structure}$ denotes that attributes and their connection in a summary are known, while S^{value} denotes the values of the attributes sought.

A Type 1 summary may be easily derived by a simple extension of fuzzy querying. The user has to construct a query, i.e. a candidate summary, and it has to be determined what is the fraction of rows matching this query and what linguistic quantifier best denotes this fraction. A Type 2 summary is a straightforward extension of Type 1. A Type 3 summary requires much more effort as it boils down to the determination of typical or exceptional, depending on the quantifier, values of an attribute. A Type 4 summary is meant to find typical (exceptional) values for some, possibly fuzzy, subsets of rows. A Type 5 summary represent the most general form considered here: fuzzy rules describing dependencies between specific values of

particular attributes. Type 1 and Type 3 summaries have been implemented as an extension to Kacprzyk and Zadrożny's [10] FQUERY for Access.

Two approaches to Type 5 summaries generation have been proposed. First, a subset of such summaries may be obtained by analogy with association rules concept and employing their efficient algorithms (cf. Borgelt and Kruse [2]). Second, genetic algorithms may be used to search the space of summaries (cf. George and Srikant [5], Kacprzyk and Strykowski [6]).

4 Remarks on Some Implementations

In this section we will briefly show for illustration some linguistic summaries derived in various domains for various purposes: linguistic summarization of corporate innovation data and linguistic summarization of Web server logs. Basically, the first example concerns static data, and the second is mainly concerned with static data but extends the analysis to dynamic data. As we will see, various protoforms are employed in those examples but, in general, they are highly comprehensible and can be well understood by domain experts.

4.1 Linguistic Summaries of Data on the Innovativeness of Companies

In the first example the purpose was to develop a human consistent, linguistic summarization based tool for the analysis of data related to the innovativeness of Polish companies (cf. Baczko, Kacprzyk and Zadrożny [1]). The values of each attribute were described by three linguistic terms: low, medium and high. The definition of linguistic terms was supported by FQUERY for Access. The linguistic quantifier "most" was used in the generated summaries. The set of transformed data was processed by AprioriTID in Borgelt's implementation (cf. `http://www.borgelt.net/apriori.html`).

We obtained a lot of very interesting linguistic summaries exemplified by:

> "Most companies having high net revenues from sales and equivalent in 2004 had high total assets in 2004"
>
> "Most of the companies having at least a few points (scores) for their RTD related activities in 2006 had also some points for that in 2005"
>
> "Most companies having some points related to patents registered in 2006 AND some points for their RTD related activities in 2005 had also some points for RTD related activities in 2006"

Thus, in general, companies being active in the RTD field in 2005 did not necessarily continue to do so in 2006. However, those with some patents in 2006 usually also had RTD related activities in 2006.

Notice that the very structure of the linguistic summaries, i.e. their underlying protoforms, have an extremely high degree of comprehensibility for the domain experts specializing in innovations, economics, etc.

4.2 *Linguistic Summaries of Web Server Logs*

A Web server log file may be directly interpreted as a table of data with the columns corresponding to the fields listed in Table 2 and the rows corresponding to the requests. On the other hand, the content of a log file may be naturally viewed as a time series type of data as each request is time stamped; such type of data reflects an inherent dynamics. In this section we will discuss various linguistic summaries that may be derived using that type of data as proposed by Zadrożny and Kacprzyk [30].

Table 2 Content of the web server log file

Field no.	Content
1	the requesting computer name or IP address
2	the username of the user triggering the request
3	the user authentication data
4	the date and time of the request
5	the HTTP command related to the request which includes the path to the requested file
6	the status of the request
7	the number of bytes transferred as a result of the request
8	the software used to issue the request

The fourth field (cf. Table 2), i.e. a timestamp that is accompanying each recorded request, plays a special role as it may be used to form summaries like: "Most of large files requests take place on Thursdays". Here the time (instant) is treated as any other nominal attribute. Such summaries will be referred to as *static* as they concern time (Thursday) but in a static sense.

On the other hand the time series perspective of the data may imply the following summary: "Recently, most of decreasing trends are very short". We will refer to such summaries as *dynamic* as they concern time but in a dynamic sense taking explicitly into account what has been happening over some time period, here recently. In the following subsections we study both types of linguistic summaries in a more detail.

Static Summaries

We denote by Y the set of all analyzed requests to a Web server, and we describe a request by the attributes given in Table 3 which directly correspond to the fields listed in Table 2 or are extracted from them. Obviously, the extraction of other attributes is possible and may lead to interesting summaries, too, but this will not be discussed here.

We can distinguish *simple summaries*, where the qualifier R is absent. These may be exemplified by:

Table 3 Attributes of the requests used for their linguistic summarization

Attribute name	Description
domain	Internet domain extracted from the requesting computer name (if given)
hour	an hour the request arrived; extracted from the date and time of the request
day of the month	as above
day of the week	as above
month	as above
filename	the name of the requested file, including the full path, extracted from the HTTP command
extension	the extension of the requested file extracted as above
status	the status of the request
failure	=1 if status code is of 4xx or 5xx form and =0 otherwise
success	=1 if status code is of 2xx form and =0 otherwise
size	the number of bytes transferred as a result of the request
agent	the name of the browser used to issue the request (name for major browsers, ”other” otherwise)

Most of the requests come from the Opera browser

or

Almost all requested files are *small*

Here “most” and “almost all” are linguistic quantifiers and the summarizers are “browser is Opera” and “size is small”, respectively. These are fairly simple summaries which may be deduced while looking at an appropriate tabular or graphical report produced by a popular Web log analysis software. This is particularly true in the first case.

The use of a linguistic term (“small”) in the second of these summaries is relevant from our point of view. Assuming that the linguistic terms are calibrated according to the particular reporting needs, the use of such summaries provides for a highly compressed, easily comprehensible presentation of some features of the Web server access data.

More interesting may be *extended summaries*, exemplified by:

Almost all failures concern files with an extension “ppt”

or

Most of the requests concerning *large* files happen in the *evening*

Here “most” and “almost all” are the linguistic quantifiers, the summarizers are “extension is ‘ppt”’ and “hour is in the evening”, respectively, and the qualifiers are “failure is 1” and “size is large”. The first summary may indicate that the maintenance of the archive of the Powerpoint presentations should be carried out more carefully. The second may suggest that the large reports that the company makes available at its Web server should be updated, if possible, in the afternoon rather than in the morning to provide useful and timely information.

Dynamic Summaries

In a series of our previous papers, which culminated in Kacprzyk, Wilbik and Zadrożny [19, 20], and Wilbik and Kacprzyk [23], we proposed to apply linguistic summaries to time series data. The linguistic summaries are used to describe in a human consistent way how trends concerning a selected numerical attribute evolve over time, how long some types of behavior last, how rapid changes are, etc.

We deal with a numerical attribute such as the size of the requested files or the number of the requests. These are aggregated over a uniformly spaced time moments, e.g., hours or days. Then the (partial) *trends* in such a data are identified as linear segments in a piecewise linear approximation of a time function obtained. Such an approximation may be obtained using various methods. These are clearly partial trends as a global trend in a time series concerns the entire time span of the time series, and there also may be trends that concern parts of the entire time span, but more than a particular window taken into account while extracting partial trends by using the Sklansky and Gonzalez algorithm.

Table 4 Attributes derived for the purposes of dynamic linguistic summaries

Name	Linguistic terms
dynamics of change	*decreasing*, *slowly increasing*,...
duration	*long*, *short*,...
variability	*high*, *small*,...

The *dynamics of change*, meant here as the speed of changes, is expressed by the slope (angle) of a line segment. In the linguistic summaries the following linguistic terms are used in reference to this attribute: “quickly decreasing”, “decreasing”, “slowly decreasing”, “constant”, “slowly increasing”, “increasing” and “quickly increasing”.

In fact, each term represents a fuzzy granule of directions. The user may define a membership functions of particular linguistic terms depending on his or her needs.

Duration corresponds to the length in time units of a single trend. In the summaries it is described by linguistic terms exemplified by “long”, “short” etc.

Variability is here a measure of how “spread out” are actual data points around approximating them line segment.

Again this attribute is treated as a linguistic variable and expressed using linguistic values (labels) such as "high", "low", etc.

Basically, in case of dynamic summaries we have distinguished simple and extended forms of them, as previously depending on the absence or presence of a qualifier. Moreover, we have introduced another classification of dynamic summaries into the *frequency based* and *duration based*. The former class comprises those summaries which describe the partial trends using just their attributes listed in Table 4. Summaries of the latter class explicitly exploit the existence of the time scale (duration) inherent in the data set. The examples of those summaries are given below.

Let us start with a *simple frequency based summaries* exemplified by

Most of the trends concerning the number of requests are *decreasing*

Thus here we assume that the entity which is measured over time is the number of requests. "Most" is the linguistic quantifier and *decreasing* for the "dynamics of change" is the summarizer. Let us assume that the access data are aggregated day by day and the log file covers several months. Then such a summary indicates a steady decline in the number of requests served by the Web server. Still the fact that such a summary is true does not exclude the possibility that there are a few increasing trends that are quite long, due to the existence of the "most" quantifier.

Even stronger an indication of a request rate decline is provided by the following *simple duration based summary*:

Trends concerning the number of requests that took *most time* are *slowly increasing*

Here, again, some increasing trends are not excluded but they are short in terms of the total time they last altogether. Here, again,"most" is a linguistic quantifier but this time referring to the time covered by the trends rather than to their number. The summarizer is "dynamics of change is slowly increasing".

The extended frequency based summaries may be exemplified by:

Most of *increasing* trends concerning the number of requests are of *high* variability

This indicates that if there is a growth of the requests rate, then usually the number of requests fluctuates seriously. The linguistic quantifier, summarizer and qualifier are "most", "variability is high" and "dynamics of change is increasing", respectively.

Finally, we may consider the extended duration based summaries exemplified by:

Increasing trends concerning the total size of requested files, that took *most* of the time, are *very long*

and this summary states that among increasing trends concerning the total size of requested files, the predominant ones, i.e., taking most of the time, are those which are very short.

It can be evidently seen that the linguistic summaries obtained do provide much highly valuable information, for both the maintenance, design and improvement of the Web servers. Moreover, the form of the linguistic summaries is extremely comprehensible to the human user, too. And, again, the use of various protoforms,

which lead to various summaries, is of utmost importance as it makes it possible to emphasize elements of relevance.

5 Concluding Remarks

We have presented the essence of our approach to linguistic database summaries, equated with linguistically quantified propositions in Zadeh's sense and mined through the use of a fuzzy querying interface to a database. We have recasted the problem from the perspective of comprehensiveness of patterns derived by linguistic data summaries. The use of natural language, which was advocated by Michalski [21] in his seminal approach to the comprehensiveness of data mining and machine learning results, has provided a new quality and an exceptional human consistency and comprehensiveness. We have illustrated our analysis by two examples related to the linguistic summarization of both static and dynamic data in the area of analysis of innovativeness of companies and of Web server log files.

References

[1] Baczko, T., Kacprzyk, J., Zadrożny, S.: Towards knowledge driven individual integrated indicators of innovativeness. In: Jozefczyk, J., Orski, D. (eds.) Knowledge-Based Intelligent System Advancements: Systemic and Cybernetic Approaches, pp. 129–140. IGI Global, Hershey (2011)

[2] Borgelt, C., Kruse, R.: Induction of association rules: Apriori implementation. In: Proc. 15th Conf. on Comp. Statistics (Compstat 2002), Berlin, Germany, pp. 395–400. Physica-Verlag, Heidelberg (2002)

[3] Craven, M.W., Shavlik, J.W.: Extracting comprehensible concept representations from trained neural networks. In: Working Notes of the IJCAI 1995 Workshop on Comprehensibility in Machine Learning, Montreal, Canada, pp. 61–75 (1995)

[4] Fisch, D., Gruber, T., Sick, B.: Swiftrule: Mining comprehensible classification rules for time series analysis. IEEE Transactions on Knowledge and Data Engineering 23(5), 774–787 (2011)

[5] George, R., Srikanth, R.: Data summarization using genetic algorithms and fuzzy logic. In: Herrera, F., Verdegay, J.L. (eds.) Genetic Algorithms and Soft Computing, pp. 599–611. Physica-Verlag, Heidelberg (1996)

[6] Kacprzyk, J., Strykowski, P.: Linguistic summaries of sales data at a computer retailer: A case study. In: Proceedings of IFSA 1999, Taipei, Taiwan R.O.C, vol. 1, pp. 29–33 (1999)

[7] Kacprzyk, J., Yager, R.R.: Linguistic summaries of data using fuzzy logic. International Journal of General Systems 30, 133–154 (2001)

[8] Kacprzyk, J., Zadrożny, S.: FQUERY for Access: Fuzzy querying for a Windows-based DBMS. In: Bosc, P., Kacprzyk, J. (eds.) Fuzziness in Database Management Systems, pp. 415–433. Physica-Verlag, Heidelberg (1995)

[9] Kacprzyk, J., Zadrożny, S.: Fuzzy queries in Microsoft Access v.2. In: Proc. FUZZ-IEEE/IFES 1995, Workshop on Fuzzy Database Systems and Information Retrieval, Yokohama, Japan, pp. 61–66 (1995)

[10] Kacprzyk, J., Zadrożny, S.: On combining intelligent querying and data mining using fuzzy logic concepts. In: Bordogna, G., Pasi, G. (eds.) Recent Research Issues on the Management of Fuzziness in Databases, pp. 67–81. Physica-Verlag, Heidelberg (2000)
[11] Kacprzyk, J., Zadrożny, S.: Computing with words in intelligent database querying: Standalone and internet-based applications. Information Sciences 134, 71–109 (2001)
[12] Kacprzyk, J., Zadrożny, S.: Data mining via linguistic summaries of databases: An interactive approach. In: Ding, L. (ed.) A New Paradigm of Knowledge Engineering by Soft Computing, pp. 325–345. World Scientific, Singapore (2001)
[13] Kacprzyk, J., Zadrożny, S.: Fuzzy linguistic summaries via association rules. In: Kandel, A., Last, M., Bunke, H. (eds.) Data Mining and Computational Intelligence, pp. 115–139. Physica-Verlag, Heidelberg (2001)
[14] Kacprzyk, J., Zadrożny, S.: Linguistic summarization of data sets using association rules. In: Proc. FUZZ-IEEE 2003, St. Louis, USA, pp. 702–707 (2003)
[15] Kacprzyk, J., Zadrożny, S.: Linguistic database summaries and their protoforms: Towards natural language based knowledge discovery tools. Information Sciences 173(4), 281–304 (2005)
[16] Kacprzyk, J., Ziółkowski, A.: Database queries with fuzzy linguistic quantifiers. IEEE Transactions on Systems, Man and Cybernetics 16, 474–479 (1986)
[17] Kacprzyk, J., Zadrożny, S., Ziółkowski, A.: FQUERY III+: a 'human consistent' database querying system based on fuzzy logic with linguistic quantifiers. Information Systems 6, 443–453 (1989)
[18] Kacprzyk, J., Yager, R.R., Zadrożny, S.: A fuzzy logic based approach to linguistic summaries of databases. Int. Journal of Applied Mathematics and Computer Science 10, 813–834 (2001)
[19] Kacprzyk, J., Wilbik, A., Zadrożny, S.: Linguistic summarization of time series using a fuzzy quantifier driven aggregation. Fuzzy Sets and Systems 159, 1485–1499 (2008)
[20] Kacprzyk, J., Wilbik, A., Zadrożny, S.: An approach to the linguistic summarization of time series using a fuzzy quantifier driven aggregation. International Journal of Intelligent Systems 25(5), 411–439 (2010)
[21] Michalski, R.: A theory and methodology of inductive learning. Artificial Intelligence 20(2), 111–161 (1983)
[22] Pryke, A., Beale, R.: Interactive Comprehensible Data Mining. In: Cai, Y. (ed.) Ambient Intelligence for Scientific Discovery. LNCS (LNAI), vol. 3345, pp. 48–65. Springer, Heidelberg (2005)
[23] Wilbik, A., Kacprzyk, J.: Towards a multi-criteria analysis of linguistic summaries of time series via the measure of informativeness. International Journal of Data Analysis Techniques and Strategies 4(2), 181–204 (2012)
[24] Yager, R.R.: A new approach to the summarization of data. Information Sciences 28, 69–86 (1982)
[25] Yager, R.R.: On ordered weighted averaging operators in multicriteria decision making. IEEE Trans. on Systems, Man and Cybern 18(1), 183–190 (1988)
[26] Yager, R.R., Kacprzyk, J.: The Ordered Weighted Averaging Operators: Theory and Applications. Kluwer, Boston (1997)
[27] Zadeh, L.A.: A computational approach to fuzzy quantifiers in natural languages. Computers and Maths with Appls. 9, 149–184 (1983)
[28] Zadeh, L.A.: A prototype-centered approach to adding deduction capabilities to search engines — the concept of a protoform. BISC Seminar (2002)
[29] Zadeh, L.A., Kacprzyk, J. (ed.): Computing with Words in Information/Intelligent Systems, 1. Foundations, 2. Applications. Physica-Verlag, Heidelberg (1999)

[30] Zadrożny, S., Kacprzyk, J.: From a static to dynamic analysis of weblogs via linguistic summaries. In: Proceedings of 2011 IFSA World Congress and AFSS Congress, Surabaya, Indonesia (2011)
[31] Zadrożny, S., De Tré, G., De Caluwe, R., Kacprzyk, J.: An overview of fuzzy approaches to flexible database querying. In: Galindo, J. (ed.) Handbook of Research on Fuzzy Information Processing in Databases, pp. 34–53. Idea Group, Inc. (2008)
[32] Zhou, Z.H.: Comprehensibility of data mining algorithms. In: Wang, J. (ed.) Encyclopedia of Data Warehousing and Mining, pp. 190–195. IGI Global, Hershey (2005)

Listening to the Voice of the Customers: An Early Warning System Based on Sentiment

Carsten Lanquillon

Abstract. In a global economy, enterprises have to intelligently analyze their data asset in order to stay competitive. With the advent of the Web 2.0, there is a wealth of user generated content which contains valuable information on what customers think about available products and services: the voice of the customers is readily accessible. Listing to and understanding these data can reveal valuable customer insights especially for product quality assessment, improvement and innovation as well as marketing. This report focuses on the use of state-of-the-art text mining techniques for identifying and monitoring the sentiment of customer feedback on Web 2.0 channels over time such as to alert enterprise users to significant increases in negative sentiment as an early indicator of inferior or degrading product quality.

1 Introduction

With the advent of the Web 2.0 and mobile technologies, social media have become an integral part of our society. Social media platforms such as blogs, microblogging, discussion forums, review sites and social networks are used to easily create and share the so-called user generated content [11].

Whether or not an enterprise has decided to use social media channels as a part of its communication strategy, existing and potential customers are likely to share the experience they made with the enterprise's products or services. Obviously, they did so even before social media platforms became popular. However, while at that time information was shared rather locally among personal contacts, information can now be shared quickly and globally. Hence, the reach of the word of mouth communication is drastically enlarged. Social media platforms have become the most influential source of information prior to buying.

Carsten Lanquillon
Heilbronn University, 74081 Heilbronn, Germany
e-mail: carsten.lanquillon@hs-heilbronn.de

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 223–235.
springerlink.com

By listening to and understanding the voice of their customers, enterprises have the possibility to efficiently gain immediate costumer insights which can be used in particular for market research, product improvement and innovation, product quality assessment as well as marketing. As such it is part of an emerging discipline referred to as *social media analytics* which attracts researchers and practitioners alike. This report focuses on state-of-the-art text mining techniques for identifying and monitoring the sentiment orientation of customer feedback on Web 2.0 channels over time such as to alert enterprise users to significant increases in negative sentiment orientation as an early indicator of inferior or degrading product quality.

At the core of social media analytics is the field known as opinion mining or sentiment analysis. The following section introduces this field and sets out the relevant tasks for this report. Subsequently, state-of-the-art text mining techniques tailored for sentiment analysis are presented. Then, the architecture of the early warning system together with its core components is described with a focus on monitoring sentiment orientation over time. The conclusion provides a brief discussion on the solutions presented and gives prospects for future research.

2 Opinion Mining and Sentiment Analysis

Liu and Zhang [19] define sentiment analysis or opinion mining as "the computational study of people's opinions, appraisals, attitudes and emotions toward entities, individuals, topics and their attributes expressed." As pointed out in the previous section, opinions play a key role in our everyday life in general and in particular in decision making for both individuals (private consumers) and organizations (business). Currently, the dominant source of opinions in user generated content is still textual data which is also the focus of this report. In the past decade, sentiment analysis has been studied extensively. The following generic definition of the underlying tasks enables a precise classification of the envisioned application's components.

The general goal of opinion mining or sentiment analysis is to turn unstructured or semi-structured text collections into structured data, which is subsequently amenable to further qualitative and quantitative analysis and visualization. In this context, an opinion can be defined as a quintuple (t, h, e, a, s), where

- t is the time when the opinion was expressed,
- h is the opinion holder,
- e is a target entity,
- a is an aspect (feature) of the entity e, and
- $s = s(t, h, e, a)$ is the supposedly unique sentiment orientation (polarity) of the opinion on aspect a of entity e expressed by opinion holder h at time t [18].

These elements are considered essential for characterizing an opinion [18]. Yet, note that further elements may be helpful or required for specific applications such as location information, attributes characterizing the opinion holder, or the usefulness of an opinion which is often available especially in the domain of review sites.

Based on this definition, the general objective of opinion mining can formally be phrased as the discovery of all opinion-quintuples from a set of documents and naturally be split into five subtasks:

- Date and time extraction
- Opinion holder extraction
- Entity identification
- Aspect identification
- Sentiment orientation (polarity) detection

Eventually, the five elements are combined to construct an opinion-quintuple. While all tasks can already be very challenging on their own, it is also not trivial to ensure that the elements within extracted opinion tuples genuinely belong together [18].

The detection of the sentiment orientation is key to all sentiment analyses. In fact, many research efforts focus solely on this subtask. Depending on the task or application at hand, the remaining four elements may not be relevant or not even applicable. For example, in many applications the opinion holder need not be identified or may conveniently be assumed to be the only author of a document. The time when an opinion was expressed is mandatory for applications that visualize or monitor opinions over time. Lastly, the applicability of the entity and aspect strongly depends on the localization of sentiment within a document:

Document-level sentiment: The primary task is to determine the sentiment orientation of an entire document while ignoring the remaining opinion elements. This may be helpful when the task is to extract a total (summary) sentiment of a document. Yet, most documents except for very small documents such as *tweets* typically contain more than one opinion of often diverse polarity. Nevertheless, much research has been devoted to this level of sentiment analysis.

Sentence or clause-level sentiment: This perspective comes closer to individual opinions with regard to a certain entity and aspect. And for reasons of simplicity it is often assumed that a sentence contains a single opinion from an individual opinion holder. Obviously, a sentence may contain sentiment polarity on several entities and aspects as illustrated by the simple sentence "The car is great, but the gas consumption is way too high." Looking at clauses rather than sentences may ease but not completely solve this problem. Yet, it resembles a sound compromise between the coarse-grained document-level and the fine-grained entity and aspect-level of sentiment analysis. In addition to polarity detection, opinion topics might be extracted as specific entity-aspect-combinations from each opinionated sentence or clause. Often, data selection prior to opinion mining is parameterized such that only documents relevant for a selected entity are analyzed. In this case, the extracted topics correspond to the aspect element.

Entity and aspect-level sentiment: Detecting each single opinion with its elements as defined above is the most precise but also most difficult endeavor. All entities with their relevant aspects in combination with the corresponding sentiment orientations have to be determined within a document. Often it is assumed that all opinions within a document are from the same opinion holder but obviously this need not be true.

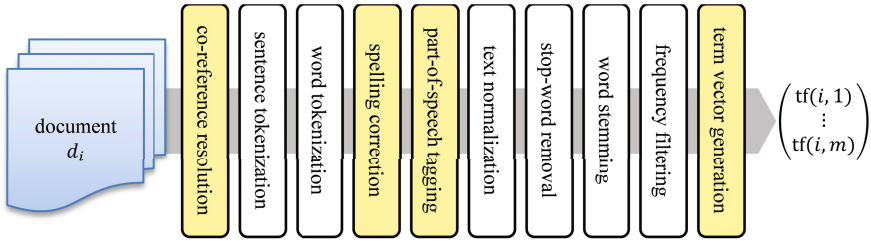


Fig. 1 The text preprocessing process transforms documents into sentence-based term vectors

In the following we focus on sentence-level sentiment analysis for documents which are gathered for an entity of choice. Further, we assume that the task of extracting the author (opinion holder) as well as the date and time when the opinion (document) was expressed are solved for the selected sources. Thus, the subtasks to be solved and discussed below by means of text mining technologies are topic identification and sentiment orientation detection within individual sentences.

3 Text Mining Technologies for Sentiment Analysis

This section describes the core text mining technologies for sentiment orientation detection which will be treated as a text classification task and topic identification by means of text clustering. General issues regarding text preprocessing and representation are relevant for both tasks and covered first. In the following we assume that a collection of plain text documents relevant for a specific entity of choice has been gathered. Furthermore, we do not cover the issue of multi-language processing and assume that the text collection contains documents in English only.

3.1 Text Preprocessing and Representation

Before any text mining on a collection of documents can take place, the documents need to preprocessed and transformed into a format that is suitable for analysis such as term vectors. Moreover, adequately designing the text preprocessing process is a crucial factor with regard to the quality of text mining results. Fig. 1 shows the most important steps involved in the text preprocessing process. Some of the steps may require deeper knowledge of natural language processing techniques which are beyond the scope of this report.

Sentence and word tokenization, text normalization, stop-word removal, word stemming, and frequency filtering are carried out in a typical manner [14]. Steps with less common or task-specific aspects are highlighted in Fig. 1 and motivated as follows.

Co-reference resolution: Our objective is a sentence-level sentiment analysis. This requires sentence tokenization which will take sentences out of their context

so that references between sentences will be broken. Co-reference resolution is a remedy to this problem as it attempts to replace each of possibly several terms which all reference the same entity by only one defined term [26]. For example, "Martin Winterkorn is the CEO of Volkswagen AG. *He* succeeded Bernd Pischetsrieder as CEO of Volkswagen AG in 2007" should be transformed into "Martin Winterkorn is the CEO of Volkswagen AG. *Martin Winterkorn* succeeded Bernd Pischetsrieder as CEO of Volkswagen AG in 2007" such that each sentence is more comprehensible in isolation.

Spelling correction: Contrary to many other text mining tasks, documents obtained from social media are likely to contain many spelling errors. These may hinder the following steps from producing results of acceptable quality. Thus, spelling errors should be automatically corrected if possible [5].

Part-of-speech tagging: A part-of-speech tagger assigns word class information to each word such as a word being a *noun, verb, adjective* or *adverb*. Here, this information may be utilized for task specific feature selection. In particular, sentiment orientation detection may rely rather on adjectives and adverbs whereas nouns and adverbs may be more relevant for opinion topic identification.

Term vector generation: This step aims at transforming the sequence of words (terms) from a specific unit of text into a term vector. Note that the general terminology typically refers to these basic units of text as *documents*. Yet, here the units of text are actually *sentences* or *clauses* of documents. A very common way of representing text is the *bag-of-words* approach based on word-unigrams. This approach does not take into account the order in which words occur within a document, i.e., here, within a *sentence*. Instead, the number of occurrences of each term in each sentence is counted which leads to the so-called term frequencies (tf). More sophisticated approaches might construct terms based on small sequences of words (word n-grams) e.g. to capture multi-word expressions such as "*social media monitoring*". Although the context of words is intuitively relevant, in many applications and domains, often there is no or only little improvement with regard to the quality of text mining results when using multi-word terms instead of unigram terms [25]. In part, this is due to limited text resources as opposed to the rapidly growing dimensionality which comes along with more complex term definitions. Therefore, we will use the plain unigram bag-of-words approach. Each *sentence* is represented by a vector of absolute term frequencies where each position in the vector corresponds to a specific term which is element of a particularly selected term set also referred to as the *vocabulary* V. Depending on the subsequent text mining task, the term frequencies might be further processed. Common choices are using simple binary *term presence* in a sentence rather than the absolute frequencies or using weighted frequencies, e.g. through multiplication with the so-called *inverse document frequency* (idf). This inverse frequency is based on the number of "documents" (i.e., here, sentences) in which a term occurs and takes into account that terms have more discriminating power when they occur only in a (small) subset of the "document" collection [27].

3.2 Sentiment Orientation Detection

An intuitive approach to sentiment orientation detection is to treat it as a standard (topic-based) text classification task with either two classes (*positive* and *negative* polarity) or three classes by adding a supplemental *neutral* class for no or mixed polarity. In practice, support vector machines [10], Naive Bayes classifiers [16, 20] and linear threshold models [23] are frequently applied to topic-based text classification and reported to perform very well. Yet, experiments in sentiment analysis show that sentiment classification is much harder than topic classification [25].

Why is it more difficult to classify sentiment? Topic-related documents typically contain more topic-specific terms which allow for reasonably good classification performance. In contrast, polarity often depends on the domain and the aspect of an entity. One and the same term may have opposite sentiment orientation even for different aspects of the same entity. Further, topic classification appears to be more robust with respect to the occurrences of individual terms. By contrast, individual terms such as negators or other explicit valence-shifters may very well inverse polarity. In addition, implicit valence-shifting such as the use of irony aggravates this issue [8]. Lastly, user groups on different social media platforms may have their own style of communication with contrary polarity-specific meaning. In this report, we will not deal with language-specific aspects explicitly and focus on the general capabilities of text mining approaches to reasonably detect sentiment.

3.2.1 Semi-supervised Learning Framework

As sentiment classification is strongly domain and task-specific, approaches based on text mining techniques are appealing in highly automated applications. Manually constructed rule-based approaches based on natural language processing techniques may outperform text learning approaches at selected tasks. Yet, they typically require extensive user effort which may render model construction prohibitive when many entities and aspects are to be monitored. But, since supervised learning approaches to text classification require costly labeled training data, not much help is attained with regard to reducing user effort. At this point, semi-supervised approaches which learn from few labeled documents and a large number of inexpensive and readily available unlabeled documents [13, 24] should be considered. And indeed, semi-supervised approaches have already been applied successfully to the task of sentiment orientation detection [3, 7].

Our semi-supervised framework uses a two-stage bootstrapping approach to learn an initial base classifier based on polarity-specific key words instead of labeled sentences. Subsequently, the following two steps are alternated until no changes in class assignments are recognized or a specified number of iterations is exceeded:

1. Use the base classifier to predict the class of all (unlabeled) sentences, and
2. Re-learn the classifier based on sentences classified as either positive or negative.

3.2.2 Base Classifier

The Naive Bayes classifier is known to work well as a base classifier in semi-supervised learning. Yet, we refrain from using it in this application because we expect the class priors to change significantly over time. After all, the early warning systems should detect this kind of changes in sentiment orientation. Instead, we use the Winnow approach [17] as it proved to be robust and efficient with good classification performance for sentiment analysis [9].

Winnow learns a linear-threshold classifier $H_c(d)$ for class c and sentence d as

$$H_c(d) = \sum_{t \in V} w_c(t)\chi(d,t)$$

where V is the vocabulary, $\chi(d,t) = 1$ if term t occurs in sentence d ($\mathrm{tf}(d,t) > 0$) and 0 otherwise, and $w(t)$ is the weight of term t.

Winnow predicts class c for sentence d if $H_c(d) > \theta$. Note that we learn two classifiers H_+ and H_- for the positive and negative class, respectively. A sentence will be declared to be neutral if either none or both of the classifiers fire ($H_c > \theta$).

In the learning phase, the weights are initialized by $w_c(t) = 1$ for each term $t \in V$. For each positively or negatively labeled sentence d, Winnow updates the weights if it cannot correctly classify d as follows: $w_c(t)$ *= 2 if d belongs to class c and $\chi(d,t) = 1$ and $w_c(t)$ /= 2 if d does not belong to class c and $\chi(d,t) = 1$.

3.2.3 Two-Stage Bootstrapping of Base Classifier

We take up the two-stage bootstrapping approach of [6]. Instead of hand-labeling sentences to acquire training data, we provide domain-unspecific seed key terms for each polarity class, e.g. [6]

$$K_+ = \{good,\ excellent,\ love,\ happy\}$$

for the positive class and for the negative class:

$$K_- = \{bad,\ lousy,\ terrible,\ hate,\ suck,\ unreliable\}.$$

Under the assumption that polarity-specific key terms of the same polarity tend to co-occur at the sentence level while they do not co-occur if they are of opposite orientation [6], the initial seed sets of key terms are expanded by unambiguously co-occurring terms in a first bootstrapping phase as follows. First, all frequent 2-itemsets are generated taking sentences to resemble transactions and term occurrences within a sentence as items. Then, while there exists a frequent 2-itemset $f = \{t_1, t_2\}$ which contains an element of the key term set of one polarity class, say $t_1 \in K_c$, but no element of the other class ($f \cap K_{\bar{c}} = \emptyset$), the other term t_2 is added to the respective key term set K_c.

Based on these polarity-specific key term sets, sentences are initially and unambiguously labeled as class c if they contain at least one key term of the corresponding key set K_c and none of the opposite key set. The resulting set of labeled sentences is

used to initially train the base classifier. In case no sentences can be labeled according to this bootstrapping process, the unambiguity requirement could be relaxed or, otherwise, some sentences have to be hand-labeled.

3.3 Opinion Topic Identification

The objective of this task is to uncover opinion topics from a stream of sentences which are orthogonal to the structure induced by sentiment orientation [4] There are two predominant types of approaches that lend themselves to identify hidden topic structure within a stream of sentences: topic modeling [21] and text clustering. Since sentences are on average much shorter than documents, they are likely to be dominated by a single topic each. Therefore, allowing sentences to belong to several topics is less important and we focus on uncovering a disjoint cluster structure.

3.3.1 Text Clustering

Due to their limited lengths, sentences are unlikely to contain enough information to generate meaningful topic ontologies. Thus, we focus on partitional text clustering approaches. Text clustering based on frequent itemsets provide a sound way of dealing with the very high dimensionality of text data and providing meaningful cluster descriptions [2, 28].

Following the partitional FTC approach of [2], treating sentences as transactions and term presence in a sentence as items, frequent itemsets are generated using a standard association rule algorithms like Apriori [1]. Each frequent itemset f is regarded as the (meaningful) description of a cluster candidate. The cover $\mathrm{cov}(f)$ denotes the set of all sentences that contain the terms $t \in f$. Starting with an empty set S of selected frequent itemsets, the algorithm iteratively adds frequent itemsets to S until each sentence is covered by a selected frequent itemset $s \in S$. At each iteration, that frequent itemset f whose cover has the least overlap with the already selected frequent itemsets in S is added.

3.3.2 Guiding Search towards Opinion Topic Structure

When trying to identify structure from data without any guidance such as known class labels, we may end up with any kind of structure. To guide search into the direction of meaningful opinion topics and away from sentiment-oriented structure without any kind of user feedback, we propose to make use of knowledge derived from the sentiment classification task. In an additional feature selection step prior to clustering, all terms are discarded which appear to be highly discriminative with regard to polarity [6]. In particular, this includes all polarity-specific key terms and also all terms with Winnow-based weights above a specified threshold. The remaining terms are assumed to be more relevant with regard to topic structure. In addition, it is possible to select terms based on word classes, e.g. to retain only noun groups and verbs since adjectives and adverbs are generally believed to be more relevant for sentiment orientation detection.

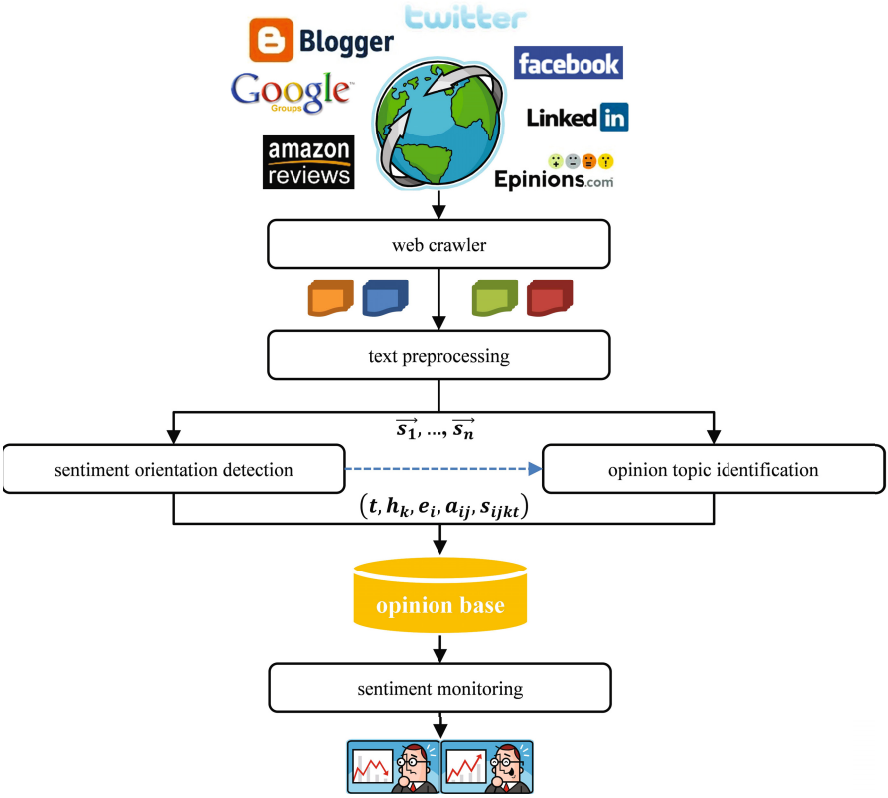


Fig. 2 System architecture of the early warning systems based on sentiment

4 Early Warning System

Fig. 2 gives an overview of the system architecture of the early warning system based on sentiment. The core components are described below.

Web Crawler

The web crawling component delivers documents from selected sources which are relevant for an entity of choice, i.e. an enterprise's product or service. In case a social media platform provides an API, accessing the user generated content for further processing might be straightforward. However, limited access to historical content can be an issue. Unfortunately, many relevant social media platforms do not provide API access to the user generated content. If tolerated by the platform provider, web crawlers and scraping tools have to be used. To deliver high quality results, i.e. the extracted opinionated text together with the opinion holder (author) as well as the date and time when the opinion was created, specific information extraction components might have to be developed for each social media source

anew unless this information extraction is treated as a separate learning task. Note that we assume that there are no legal objections such as copyright issues against extracting, storing and analyzing opinions form the sources of choice.

Text Mining Components

The components for text preprocessing, sentiment orientation detection and opinion topic identification have been described in depth in Sec. 3. Note the dotted line between the latter two components which indicates that topic identification requires information from the sentiment orientation detection component.

Opinion Base

The main objective of opinion mining is to structure the unstructured user generated content. As described above, textual documents are tokenized into sentences from which an opinion-quintuple is derived if possible. The resulting opinion quintuples together with context information such as the source system, the original source document and the URL are stored in a database which allows easy access to the opinionated content for further analysis and visualization.

Sentiment Monitoring

In order to detect changes with regard to the fraction of negative sentiment within the stream of opinionated sentences, this component applies techniques from the field of statistical quality control [22]. The key difficulty is to distinguish between chance causes of variation and causes which can be assigned to changes in sentiment orientation. Generally, change is suspected if an observed measure is out of specific control limits, i.e. if it is too many standard deviations above or below its expected value under the assumption of a stable process. Here, a variant of the well-known Shewhart control chart is deployed which has been proven to be successful in various change detection tasks within machine learning applications [12, 15].

In particular, the stream of documents is split into batches on a daily, weekly or monthly basis depending on its volume. We choose to monitor the absolute number of sentences with negative sentiment orientation per batch and opinion topic as well as for the entire set of sentences by means of a so-called np-chart. The advantage of this approach over the classical p-chart for fractions of negative opinions (resembling the fraction defective in quality control parlance) is that the users are not only informed about changes in sentiment but can also get an impression of changes with regard to opinion volume in total.

For either a certain topic or the total number of sentences, let n_t denote the number of sentences and p_t denote the fraction of negatively oriented sentiment in batch t. The corresponding expected fraction $\bar{p}$ of negative polarity under the assumption of a stable public opinion is either determined based on the initially available collection of documents which is also used during the text learning steps or it is provided by the users based on experience.

As we are only interested in significant increases in negative sentiment, we deploy only upper control limits. We assume that the random variable indicating whether or not an opinion has negative sentiment follows a binominal distribution. Hence, the upper warning limit is set at $n_t\bar{p} + 2\sqrt{n_t\bar{p}(1-\bar{p})}$ and the upper action limit is at $n_t\bar{p} + 3\sqrt{n_t\bar{p}(1-\bar{p})}$. This corresponds to two and, respectively, three standard deviations above the expected value. Finally, whenever p_t exceeds these upper control limits, an appropriate signal is issued to inform the users.

5 Conclusion

This report has demonstrated how text mining techniques can be applied to solve some of the key issues in opinion mining or sentiment analysis as a core field of the emerging discipline known as social media analytics. Since the task of sentiment orientation detection is known to be domain-specific, learning appropriate models with as little user feedback as possible is crucial for automatic monitoring systems.

Semi-supervised text learning approaches require only little user input such as some seed keywords for bootstrapping an initial base classifier for sentiment orientation detection. Yet, still better classification performance on the sentiment orientation detection task is expected when coupling text learning approaches with deeper natural language processing.

Text clustering on a feature subset restricted based on polarity-specific key terms allows for automatic identification of relevant topics which are orthogonal to sentiment orientation. The approach chosen should be further extended to incrementally deal with new documents. Detecting new hot topics from a stream of text documents may also serve as a key indicator for product quality management.

References

[1] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of 20th International Conference on Very Large Data Bases (VLDB 1994), pp. 487–499 (1994)

[2] Beil, F., Ester, M., Xu, X.: Frequent term-based text clustering. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 436–442 (2002)

[3] Dasgupta, S., Ng, V.: Mine the easy, classify the hard: A semi-supervised approach to automatic sentiment classification. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pp. 701–709 (2009)

[4] Dasgupta, S., Ng, V.: Topic-wise, sentiment-wise, or otherwise? Identifying the hidden dimension for unsupervised text classification. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pp. 580–589 (2009b)

[5] Dey, L., Haque, S.K.M.: Opinion mining from noisy text data. In: Proceedings of the Second Workshop on Analytics for Noisy Unstructured Text Data (AND 2008), pp. 83–90. ACM, New York (2008)

[6] Gamon, M., Aue, A.: Automatic identification of sentiment vocabulary: exploiting low association with known sentiment terms. In: Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing, pp. 57–64 (2005)
[7] Gamon, M., Aue, A., Corston-Oliver, S., Ringger, E.: Pulse: Mining Customer Opinions from Free Text. In: Famili, A.F., Kok, J.N., Peña, J.M., Siebes, A., Feelders, A. (eds.) IDA 2005. LNCS, vol. 3646, pp. 121–132. Springer, Heidelberg (2005)
[8] Hao, Y., Veale, T.: An ironic fist in a velvet glove: Creative mis-representation in the construction of ironic similes. Minds and Machines 20(4), 635–650 (2010)
[9] Hurst, M.F., Nigam, K.: Retrieving topical sentiments from online document collections. In: Document Recognition and Retrieval XI, pp. 27–34 (2004)
[10] Joachims, T.: Text categorization with suport vector machines: Learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 137–142. Springer, Heidelberg (1998)
[11] Kaplan, A., Haenlein, M.: Users of the world, unite! the challenges and opportunities of social media. Business Horizons 53(1), 59–68 (2010)
[12] Lanquillon, C.: Dynamic aspects in neural classification. International Journal Intelligent Systems in Accounting, Finance and Management 8(4), 281–296 (1999)
[13] Lanquillon, C.: Partially Supervised Text Classification: Combining Labeled and Unlabeled Documents Using an EM-Like Scheme. In: Lopez de Mantaras, R., Plaza, E. (eds.) ECML 2000. LNCS (LNAI), vol. 1810, pp. 229–237. Springer, Heidelberg (2000)
[14] Lanquillon, C.: Enhancing text classification to improve information filtering. PhD thesis, University of Magdeburg (2001)
[15] Lanquillon, C., Renz, I.: Adaptive information filtering: detecting changes in text streams. In: Proceedings of the Eighth International Conference on Information and Knowledge Management (CIKM 1999), pp. 538–544 (1999)
[16] Lewis, D.D.: Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 4–15. Springer, Heidelberg (1998)
[17] Littlestone, N.: Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine Learning 2(4), 285–318 (1987)
[18] Liu, B.: Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data, 2nd edn. Data-Centric Systems and Applications. Springer (2011)
[19] Liu, B., Zhang, L.: A survey of opinion mining and sentiment analysis. In: Aggarwal, C.C., Zhai, C. (eds.) Mining Text Data, pp. 415–463. Springer (2012)
[20] McCallum, A., Nigam, K.: A comparision of event models for Naive Bayes text classification. In: Proceedings of the AAAI 1998 Workshop on Machine Learning for Text Categorization, pp. 41–48 (1998)
[21] Mei, Q., Ling, X., Wondra, M., Su, H., Zhai, C.: Topic sentiment mixture: modeling facets and opinions in weblogs. In: Proceeding of the 16th International Conference on World Wide Web (WWW 2007), pp. 171–180 (2007)
[22] Montgomery, D.C.: Introduction to Statistical Quality Control, 3rd edn. John Wiley & Sons (1997)
[23] Nigam, K., Lafferty, J., McCallum, A.: Using maximum entropy for text classification. In: Proceedings of the IJCAI 1999 Workshop on Machine Learning for Information Filtering, pp. 61–67 (1999)
[24] Nigam, K., McCallum, A., Thrun, S., Mitchell, T.M.: Text classification from labeled and unlabeled documents using EM. Machine Learning 39(2/3), 103–134 (2000)

[25] Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? sentiment classification using machine learning techniques. In: Proceedings of Conference on Empirical Methods in Natural Language Processing, pp. 79–86 (2002)
[26] Rahman, A., Ng, V.: Supervised models for coreference resolution. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pp. 968–977 (2009)
[27] van Rijsbergen, C.J.: Information Retrieval. Butterworth (1979)
[28] Zhang, W., Yoshida, T., Tang, X., Wang, Q.: Text clustering using frequent. Knowledge-Based Systems 23(5), 379–388 (2010)

Part V
Applications

Computational Intelligence to Recognize Animal Vocalization and Diagnose Animal Health Status

Gerhard Jahns

Abstract. Without information there is no life. In the course of evolution, nature made use of nearly every physical principle to enable organisms to gain information from and about their environment, and to affect their environment. In animal realm, sound is one of the most prominent communication means, suited for long and close distances. Acoustic monitoring of farm animals may serve as an efficient management tool to enhance animal health, welfare, and farm efficiency, and in general as a useful tool in animal ethology. The final goal is a call-recognizer to identify the meaning of sounds issued by animals. Such call-recognizer must be able to recognize the meaning of calls, independent from the individual animal and a more or less noisy environment. As a probabilistic method during the learning or training phase, feature vectors from known calls are calculated. From feature vectors of calls with the same meaning reference patterns are built and stored. To recognize a call it has to be calculated in the same way. The system then determines the reference pattern that is most similar to the pattern to be recognized and outputs the meaning. Despite the vocabulary size and complexity of human speech, which is unique in animal realm, sound production and reception have several commonalities among vertebrates. This encourages to adapt methods and experiences from speech recognition in order to recognize animal vocalization and sounds. In speech recognition, double stochastic processes, such as Hidden Markov Models (HMMs), have proven to be very efficient. They were applied here to recognize the meaning of different animal calls in two studies, using utterances of cows as an example, and to diagnose pathological coughing of pigs. The results revealed that probabilistic methods like HMMs are well suited for monitoring animals by identifying the meaning of their vocalization and to diagnose their health status.

Gerhard Jahns
Grasgarten 9, 38176 Wendeburg/Bortfeld, Germany
e-mail: g.jahns@tu-bs.de

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 239–249.
springerlink.com

1 Introduction

Communication is a main key to survival. In the course of evolution, organisms have evolved a diversity of modalities for communication [5]. Nature has utilized nearly every physical principle, e.g. sound and vibration, electromagnetic waves (vision), chemicals (odor) and even electrical fields. Among these, sound is one of the most prominent means to convey information over long distances as well as in close vicinity, despite obstacles or the need for visual contact. These characteristics make sound analysis one of the most suitable approaches for monitoring animals. The advantage of acoustic monitoring is that no physical contact is needed. So acoustic monitoring does not interfere with the natural behavior of animals. Moreover, one system would be sufficient to continuously monitor a group of animals. An understanding of the information uttered by animals could provide farmers an efficient management tool to enhance animal health, welfare, and farm efficiency. The costs of the required hardware are quite low, and the performance of today's common PCs is more than sufficient for this purpose. The methods illustrated in this paper can be expected to be applied for species other than pigs and cows as well and be a useful tool for ethology research.

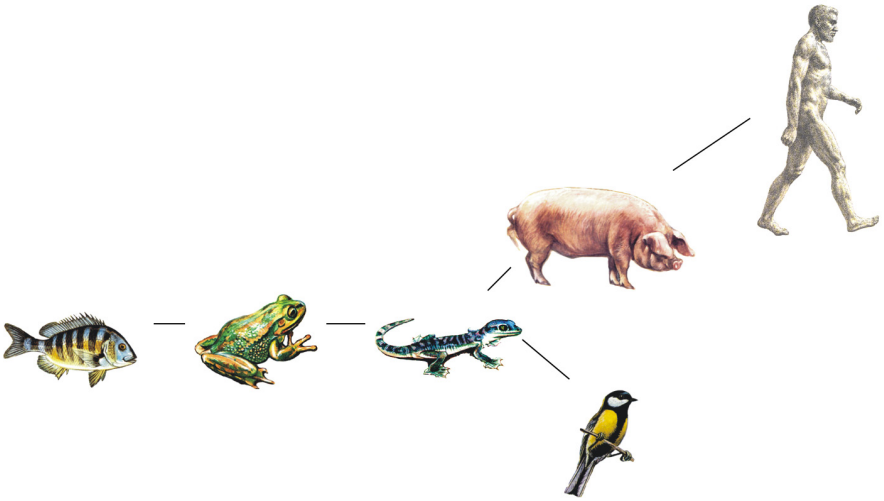


Fig. 1 Phylogenetic of vertebrates

The phylogeny of the vocal tract for sound production [12] and the acoustic organs for sound reception [1] show many commonalities. This suggests methods that have proven themselves in speech recognition, to adapt to recognize and analyze

animal utterances[1]. Figure 1 illustrates and reminds that the phylogeny of vertebrates reveals the same principles of sound production and reception [12, 9]. The recognition of *utterances* in cows is one example given below. The diagnosis of respiratory diseases of pigs by analyzing their cough is another example demonstrated here.

In the following examples it is shown, how to gain information from sounds uttered by animals. The final goal is a call-recognizer, a device that automatically transcribes animal utterances to identify their state and condition. However this needs more. Such a call-recognizer must not only be able to recognize species specific calls or sound, independent from the individual animal, it also must perform reliably in more or less noisy environments. This capability would allow farmers to monitor farm animals continuously without additional workload. Up to now, no appropriate software has been developed. Moreover, the hardware for such a call-recognizer is also appropriate to monitor technical equipment in a barn such as conveyors, augers, feeders, etc. Methods to monitor machinery by sound are state of the art, to detect damage of ball bearings, leaks in pipelines, knock sensors in cars and so on. While call analysis, based on power spectrum density (PSD), has proved to be sufficient to identify different cows individually [8], it is not sufficient to identify the meaning of their calls. It is more sophisticated to recognize the meaning of species specific calls independent of the individual animal. It may be compared with speaker independent word spotting in speech recognition. Hidden Markov Models (HMMs), which statistically model acoustic patterns, have proven very efficient for this purpose in speech recognition. To reveal the meaning of utterances in cows [7] the Hidden Markov Model Toolkit (HTK) was used. The HTK is a comprehensive set of tools and programs, developed by Young et al. [17]. To diagnose respiratory infections of pigs [4] the software toolbox from Murphy [11] was applied. Both examples revealed that HMMs are well suited for animal monitoring. To achieve sufficient results, a careful feature extraction and fine-tuning of the HMM is required. Indispensable for probabilistic methods is a large and reliable database (data corpus).

[1] The development of knowledge about the animal calls from ancient Greece up to the middle of last century is depicted by Körting in his dissertation [9]. Amongst others he concisely displays, that the ancient Greeks, among them Aristotle, Plato, Democritus, etc. had no doubts that animals are endowed with a soul, and therefore were capable to communicate and express their feelings and emotions by utterances. Thus Phorphyrius (233–301 aD) states, besides other examples, that a good cowherd would very well be able to recognize by the calls of a cow, if they have hunger or thirst, are tired, or are sexually aroused or call for their calves. With the spread of Christianity, this knowledge became forgotten. Because only humans, as the crown of creation, were regarded as the only soulful beings. Thusly still Descartes (1596–1650) regarded animals only as a *mechanical automata*. It was not until the middle of the 19th century by Darwinian (Darwin 1809–1882) theory of evolution which slowly changed this view. Today it is accepted in science, that living organisms communicate in various ways by exchanging signs and information [16].

2 Motivation and Objectives

In small farms, farmers know their animals *personally* leading to an individual, species-appropriate, and efficient animal husbandry. On large farms, where maintenance of the animals is not always in the hands of professionals, this cannot be guaranteed. Therefore, in the scope of *precision livestock farming*, big efforts have been made to monitor farm animals automatically. Established or proposed systems are expensive. Also, since they need to be attached to them or force the animals in a certain position, they may perturb the natural behavior of the animals. They also perform intermissive, which means that data transmission takes place only at particular times a day, e.g. during milking.

The diagnosis of respiratory infections in pig fattening is a serious economic issue. Respiratory infections in pig fattening farms cause losses of millions of Euros [6] respectively Dollars [15] per year. These economic losses are attributed less to total losses due to the death of pigs, but much more to costs for treatment and indirect costs caused by diminished mast und breeding results. The earlier infected animals are detected, the more promising treatments and means to reduce the risk of infection are. For diagnosis, it is necessary to distinguish between cough made by healthy pigs, simple throat cleaning, and cough by infected pigs.

The objectives were to develop monitoring software, for being able to recognize and understand utterances of cows and to diagnose respiratory infections in pigs. It is also mandatory that this has to be realized without dependence on the particular animal which utters the sound. The latter is important, because if the system is not animal independent, it would be necessary to train the system to the individual *pronunciation* of each single animal to be monitored. The resulting workload would be absolutely unacceptable for practical purposes in agriculture. To utilize such a system in practice e.g. on farms, it is required to spot only utterances of distinct meaning and ignore all the other utterances and noise in the environment. This problem has not been solved yet.

3 Peculiarities of Animal Utterances

From daily experience with human speech we know that *pronunciation* of words show inter and intra individual variations. Words are never pronounced exactly the same, even if spoken by the same speaker. And it is even more different, when the word is spoken by different speakers. The same holds true for utterances of animals. Any utterance is a non-stationary signal, variable not only in frequency but also in duration. The variation of utterances in time is, as in speech recognition, a main challenge. Many efforts have been made in speech recognition to overcome this problem, e.g., template-based approaches with dynamic time warping [13, 2]. Hidden Markov Models (HMMs) [14, 17] have proven efficient in speech recognition. They are applied here. The results [4, 7] revealed that HMMs are well suited

for call recognition and to distinguish between cough in infected pigs and cough in healthy pigs (simple throat cleaning)[2].

4 Data

The data base is the link to reality. A reliable data corpus is the indispensable basis for any call recognition. Every species has its own call repertoire and therefore needs its own data corpus. Calls which are not part of the data corpus cannot be recognized. However, to establish a data corpus of animal utterances is very elaborate and time consuming, because animals cannot be asked to elicit an utterance with a certain meaning. So the meaning of utterances can only be disclosed by observation or by provoking the utterance by providing a certain situation. To cope with inter- and intra-individual variation the data corpus cannot be large enough.

Utterances from cows were recorded in the same environment in the former Bundesforschungsanstalt für Landwirtschaft (FAL) in Braunschweig, Germany. 688 recorded and manually labeled *moos* were collected for seven different meanings of utterances [7]. The records of coughing pigs were provided by the courtesy of the Department of Biosystems, Measure, Model & Manage Bioresponse (M3-Biores), Catholic University of Leuven in Belgium. For more details see [3, 10]. For training, validating and final testing, 232 cough records caused by *Pasteurella* infection, 160 cough records caused by *Actinobacillus* infection, and 149 cough records from healthy pigs, artificially elicit by citric acid, were available. In general, caution is demanded when recordings are made in different environments, because this comprises the risk that the system is trained to recognize the different acoustic environment instead the different utterances of the animals. Both data corpi were manually labeled. 70% of each data type was used for training, whereas 30% was used for validation of the recognition rate.

5 Method

Figure 2 portrays the signal flow of an utterance. According to the emotional state or condition of an animal (W), its sensory system triggers a call or call sequence (C). This causes vertebrates to release pressurized air from the lungs through the vocal cords, forcing it through resonance chambers – pharynx, mouth and nose cavities – and, finally, radiate the utterance from the lips or nose. If the vocal cords are tensed, the air will vibrate periodically according to the tension and a voiced sound will be produced. If the larynx is open, the vocal cords do not vibrate and, a voiceless sound is produced. In all cases the result is a sound – a pressure, respectively, density variation in the air – the call or call sequence.

Coughing is a reflex involving the whole respiratory tract, causing a short contraction of the lungs resulting in a short explosive like exhaling of air. The glottis

[2] Even a cough is not a call in the classic meaning, but for more fluently reading the word *call* will be used for both cough and cow utterances.

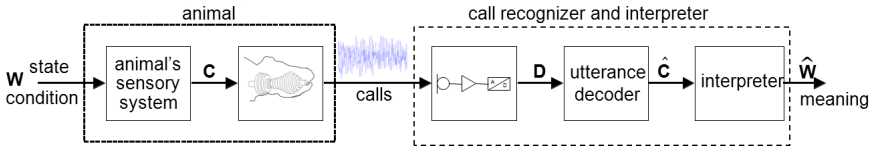


Fig. 2 Signal flow in call production and call recognition. (W) state or condition; (C) call sequence; (D) digital, one-dimensional time series; ($\hat{C}$) equivalence of the call sequence, which most likely represents the uttered calls; ($\hat{W}$) text that most likely represents the state or condition of the animal.

is open and air reaches a high speed resulting in a short characteristic noise. The primary focus was to distinguish between morbid and healthy cough, assuming that the first affects the whole respiratory tract, while the latter mainly affects the upper part, resulting in general distinguishable sounds. However, the evaluation revealed that it is even possible to distinguish between certain kinds of common infections, namely *Pasteurella* and *Actinobacillus*.

The sound event is picked up by a microphone, which changes the pressure variation into corresponding varying electrical signals, which are amplified and digitized (D). By digital signal processing, a call decoder produces an equivalent to the call or call sequence ($\hat{C}$), which most likely represents the uttered calls (C). An interpreter transcribes the call or call sequence ($\hat{C}$) into a text ($\hat{W}$) that is meaningful to humans, which most likely describes the emotional state or condition (W) of the animal.

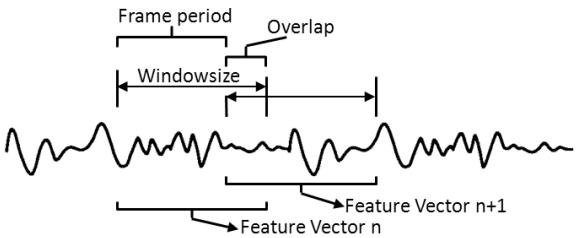


Fig. 3 Extracting feature vectors from sound wave

The first step in call recognition is to determine frequency range of the utterances of the particular species, to ensure no information is getting lost. All hard- and software has to be tuned appropriately to this frequency range. Because the sound range a species uses for communication may vary widely from infra sound (e.g. used by elephants to communicate) up to ultra sound (e.g. used by bats). The recorded signal, a continuous waveform, picked up by the microphone and amplified, results in a flow of numbers, a one-dimensional time series. From this time series, a sequence of equally spaced discrete parameters and feature vectors are calculated, respectively. These feature vectors characterize the original signal. Mathematics offers numerous

methods to extract feature vectors. Unfortunately, only some general rules can be given to choose the most appropriate features. However the requirements are contradictory. For example, the features should be chosen in a way that the variability of the features from examples belonging to the same class is reduced. On the other hand, the variability of features from examples belonging to different classes should be increased. Feature vectors should be relative imprecise to improve the robustness of the classification algorithms and to avoid over-fitting. Because each utterance is a non-stationary signal, the features were not extracted from the whole call. They were calculated within successive, overlapping, equally spaced windows (see Fig. 3).

The size of the window has to be small, thus it can be assumed that the signal is stationary within a window. Although this assumption is not strictly true, it is a reasonable approximation. The size of the window depends on the dynamic of the utterances of the species. For example in human speech, a signal within a window of about $20-30$ ms can be regarded as stationary. For the results presented below, a window of 25 ms for utterances of cows and 11 ms for pig cough was empirically chosen. To reduce disturbances at the windows' edges, they were weighted with a Hamming window.

For the cow utterances, the chosen coefficients were the Mel Frequency Cepstral Coefficients (MFCCs) with logarithmic energy and their first delta and their first acceleration coefficients. The Mel scale takes the psychoacoustic perception into account (Weber-Fechner Law) and in speech recognition it had proofed to improve recognition rates. A system for speech or call recognition comprises a training part, and a recognition part. During the training or learning phase, reference patterns respectively models were built from calls with the same meaning. As a result, each reference model represents a call of a certain meaning independent of the inter- and intra-individual variations. The more comprehensive the data corpus is, the better the call repertoire of the species is represented. For recognition, the unknown call has to be preprocessed in the same way as the calls during the learning or training phase.

Generally speaking, a Hidden Markov Model is a double stochastic process, where the states of the model are hidden [14, 17]. It is characterized by the number of states (N) and outputs (K) and defined by $\lambda = (\pi, A, B)$. The N-dimensional vector π defines the probabilities of the initial conditions. The transition probabilities from one state to another are defined by an $N \times N$-dimensional parameter matrix A. The emission probabilities, the probability that a state generates an output, are determined by an $N \times K$-dimensional matrix B.

Generally a Hidden-Markov Model allows the transition from one state to any other state. For speech- and here for call-recognition a left-right model as portrayed in Fig. 4 is used. Loops returning to the same state model the lengthening of a part of an utterance in time. Loops which skip a state model are slurring or skipping a part of an utterance. Unfortunately, there are no general rules to determine the number of states. Therefore the number of states must be determined empirically. To recognize the utterances of cows HMMs with 7 states were used for pig cough recognition HMMs with 5 states. The challenge is, to train a generic HMM to obtain the best fitting of these probabilistic parameters by samples, so that it most probably

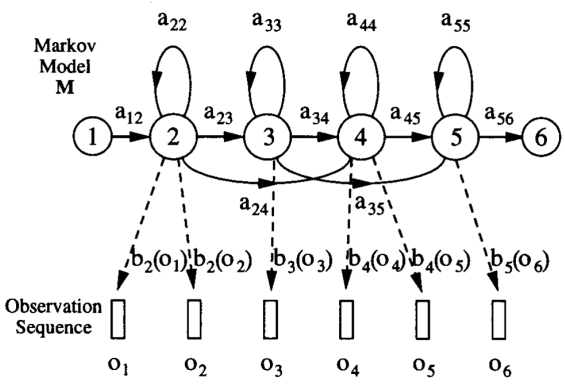


Fig. 4 Four state, five outputs left-right Hidden Markov Model [17]

represents and estimates the constituents of the utterances to be recognized. Because it is a statistical process, the number of samples should be as great as possible. For every meaning one HMM has to be trained. In particular, the model parameters of the HMM are estimated by applying the well-known Baum-Welch Algorithm. The likelihood that the HMM has produced some observation can be determined using the Viterbi Algorithm, basically a linear programming algorithm to determine the most probable sequence of states for a given observation.

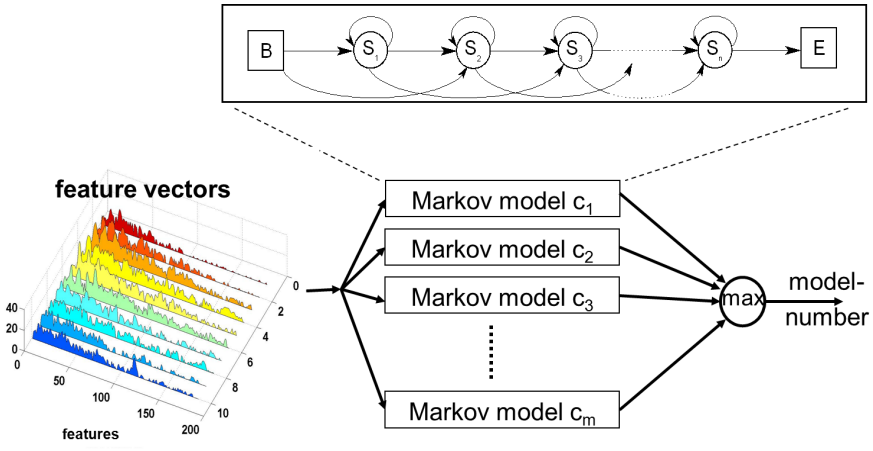


Fig. 5 Call recognition using Hidden Markov Models

Provided the HMMs representing the call repertoire are trained, Fig. 5 portrays the principle of call recognition. A model of the unknown call is computed the same way as the calls during the training phase before and then compared with the

stored reference models. The call associated with the HMM of highest likelihood is declared to be the recognized call.

6 Results

The recognition rates of different calls from cows are presented in Fig. 6. They are based on a data corpus of 688 calls. These calls were recorded from 39 cows. Seventy percent of these calls (478) were used to train the HMMs and 210 calls (30%) for validation. The diagnosis of cough reached a recognition rate of over 97%. All cough records for training as well as for recognition were filtered to reduce possible effects of the environment where they were recorded. However, the difference was minor. The few erroneously recognized unfiltered cough were two *Pasteurella* cough which were classified as *Actinobacillus* cough and one artificially generated cough which was classified as *Actinobacillus* cough. Furthermore, one filtered *Pasteurella* cough was classified as *Actinobacillus* cough and one filtered artificially generated cough was classified as *Actinobacillus* cough.

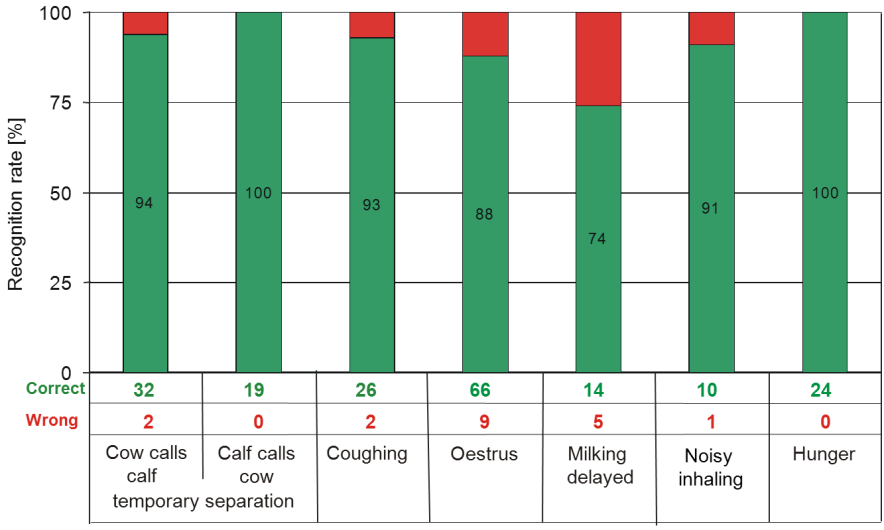


Fig. 6 Recognition rate of different calls from cows using Hidden Markov Models

7 Conclusion

From the results it can be concluded, that HMMs and methods adapted from speech recognition are suited to establish a call-recognizer, to recognize the meaning of calls from animals, here cows, and to diagnose diseases, here *Pasteurella* and *Actinobacillus* infections in pigs. However, for practical use on farms, the data corpi

have to be increased considerably. One reason is that utterances from a larger number of animals are required in order to ensure that the calls will be recognized independent of the individual animal and environment. Another reason is that more calls of equal meaning improve the recognition rates. Based on increased data corpi, the results can most probably be improved by a fine-tuning of the Hidden Markov Models. At least it must be considered that in these two studies it has only be proofed that it is possible to recognize the meaning of utterances, respectively to diagnose morbid cough. For automatic monitoring on farms, the utterances and cough have to be spotted in a continuous sound stream *buried in noise*, a problem to be solved, to establish an automatic call-recognizer.

References

[1] von Bekesy, G.: Über die mechanische Frequenzanalyse in den Schnecken verschiedener Tiere. Akustische Zeitschrift 9, 3–11 (1944)
[2] Deller, J.R., Hansen, J.H.L., Proakis, J.G.: Discrete-Time Processing of Speech Signals. IEEE Press, New York (2000)
[3] Ferrari, S., Silva, M., Guarino, M., Berckmans, D.: Analysis of cough sound for diagnosis of respiration infections in intensive pig farming. Trans. Am. Soc. Agric. Biol. Eng. 51(3), 1051–1055 (2008)
[4] Giesert, A., Balke, W., Jahns, G.: Probabilistic analysis of coughs in pigs to diagnose respiratory infections. Landbauforschung - vTI Agriculture and Forestry Research 61(3), 237–242 (2011)
[5] Hauser, M.D.: The Evolution of Communication. MIT Press, Cambridge (1997)
[6] Hoy, S.: Zu den Auswirkungen von Atemwegserkrankungen auf die Mast- und Fruchtbarkeitsleistungen der Schweine. Prakt Tierarzt 2, 121–127 (1994)
[7] Jahns, G.: Call recognition to identify cow conditions—A call-recogniser translating calls to text. Comput. Electron. Agric. 62(1), 54–58 (2008)
[8] Jahns, G., Kowalczyk, W., Walter, K.: An application of sound processing techniques for determining conditions of cows. In: Proceeding of the 4th International Workshop on Systems, Signals and Image Processing, Poznan, Poland, May 28-30 (1997)
[9] Körting, J.: Beitrag zur Entwicklung des Wissens von den Tierstimmen. PhD thesis, University of Gießen (1971)
[10] Moreaux, B., Beerens, D., Gustin, P.: Development of a cough induction test in pigs: effects of SR 48968 and enalapril. J. Vet. Pharmacol. Ther. 22(6), 387–389 (1999)
[11] Murphy, K.: Hidden Markov Model (HMM) Toolbox for Matlab. (2005), http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html (accessed on May 2, 2012)
[12] Paulsen, K.: Das Prinzip der Stimmbildung in der Wirbeltierreihe und beim Menschen. Akademische Verlagsgesellschaft, Frankfurt am Main (1967)
[13] Rabiner, L., Juang, B.: Fundamentals of Speech Recognition, Signal processing, vol. 103. Prentice Hall, Inc., Upper Saddle River (1993)
[14] Rabiner, L.R.: A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings IEEE 77(2), 257–286 (1989)

[15] Scheidt, A.: Mycoplasmal pneumonia. In: Proc. North Carolina Healthy Hogs Seminar, North Carolina, USA, November 2-5, North Carolina Swine Veterinary Group (1993), http://www.ncsu.edu/project/swine_extension/healthyhogs/book1993/scheidt1.html (accessed on May 2, 2012)
[16] Tembrok, G.: Zoosemiotik. In: Nöth, W. (ed.) Handbuch der Semiotik, 2nd edn., pp. 260–272. Metzler-Verlag, Stuttgart (2000)
[17] Young, S., Kershaw, D., Odell, J., Ollason, D., Valtchev, V., Woodland, P.: The HTK Book. Cambridge University Engineering Department, Cambridge (2006)

Applications of Intelligent Data Analysis for the Discovery of Gene Regulatory Networks

Frank Rügheimer

Abstract. The availability of cheap sequencing and measurement techniques in molecular biology sustains a rapid increase in both quantity and quality of biomedical data. But although such methods provide detailed information about gene expression levels in samples or even individual cells, these extensive data sets merely represent snapshots of system states at given times and under a limited number of conditions. Both the high data dimensionality and low throughput sample preparation present obstacles to the identification of general mechanism underlying biological functions. This contribution documents approaches that rely on Intelligent Data Analysis (IDA) to address challenges to data analysis and interpretation in Computational Biology. Several of the documented approaches were applied within a recent interdisciplinary study dedicated to the exploration of the regulatory systems in the bacterium *Bacillus subtilis*, which serves as a model for several gram-positive pathogens [6, 16].

1 Introduction

Understanding the mechanisms of regulation that allow cells to carry out a wide range of functions in varied environments has long been considered an ambitious goal in biology. Maps of regulatory interactions promise to identify targets and help limit undesired side effects in drug development, support the optimization commercial bioreactors and may even lead to new therapeutic strategies. The rapid development of experimental techniques has brought researchers within reach of that goal, and following the introduction of large scale, high throughput techniques, computational methods have become an essential element of the scientific workflow in biology. In this context the field of Computational Intelligence contributes algorithms for explorative data analysis, data integration from heterogeneous sources, decision

Frank Rügheimer
Biologie Systémique, Institut Pasteur, 75015 Paris, France
e-mail: frueghei@pasteur.fr

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 251–262.
springerlink.com

support and the validation of computationally inferred regulatory links. Although the results of such inferences are rarely sufficiently certain to stand on their own, they nevertheless guide biologists towards conducting targeted experiments that test promising candidate regulations with a high success rate.

The following section will familiarize the reader with basic concepts of gene expression and regulation in bacteria. In section 3 I will shortly present some of the most common types of data used for uncovering gene regulatory networks. The second half of the paper focuses on individual stages of a scientific process for studying gene regulation and elaborates how IDA supports each of those stages.

2 Fundamentals

2.1 From Gene to Protein to Function

Proteins constitute a large, highly varied and versatile class of molecules in living cells. They act, e.g. as enzymes, provide structural support for the cell and form specialized structures such as ion channels or membrane-bound receptors. In addition, they have functions in intra- and intercellular signaling and provide defenses against harmful chemicals, pathogens or competing organisms. In pathogens they help to bind to and extract resources from host organisms and in turn contribute to mechanisms to evade or neutralize protein-based defenses of those hosts. Furthermore, proteins are involved in elaborate mechanisms that control essential functions such as metabolism and other biochemical reactions, growth and cell division. In particular they also carry out various roles in the regulation of the cellular machinery to supply new proteins and degrade damaged or dysfunctional variants that could otherwise be harmful to the cell.

Proteins are synthesized from amino acids according to a "blueprint" of genetic information that is encoded in the cell's DNA. Genes are sections of that DNA, that code, e.g. for a protein subunit. When genes are expressed, they are initially transcribed into mobile messenger RNAs (mRNAs). The mRNA then binds to ribosomes to form a complex. In the following step the mRNA is translated at the ribosome into a corresponding chain of amino acids. These chains of amino acids subsequently fold and combine into proteins. At each of the intermediate stages additional interactions can take place that modify the product, or alter the likelihood and speed of biochemical reactions in which it partakes.

Because survival in varying environments and complex processes like cell division depend on an adapted set of proteins (proteome), cells have evolved mechanisms to regulate the level and activity of specific proteins. Bacteria, which are thought to lack some of the more elaborate mechanisms observed in eukaryotic organisms, not only serve as a test-bed for developing approaches to improve the understanding of regulation in cells, but are also interesting due to their economic significance and their potential role as human pathogens.

2.2 Genetic Regulation in Bacteria

The concentration of any given protein in living bacteria relies on a number of variable factors: the rate of synthesis of new proteins from mRNAs transcripts, the rate of degradation, and dilution due to growth. The transcription of genes is modulated by specialized proteins such as transcription factors, repressors, or terminators. These proteins recognize and bind to DNA of characteristic sequences that are located close to the regulated genes. Once bound the regulatory proteins act by recruiting other proteins and enzymes that are required for transcription or by physically blocking such elements from accessing nearby genes. Because gene transcription depends on the presence and possibly enzymatic activation of proteins and other biochemical compounds, environmental challenges often trigger a series of related transcription events that are reflected in expression data. When constructing gene regulatory networks one aims to represent a superposition of all such causal links in the gene expression patterns as a directed graph. Nodes correspond to genes or operons whereas edges represent regulatory interaction. Additional annotations are sometimes used to record details about interactions, e.g. dependence on co-factors, direction of the regulation or specific biochemical mechanisms.

2.3 Operon Model of Gene Expression

Bacterial genomes are organized in operons [11]. An operon is a group of genes that are located on a continuous section of DNA and that are transcribed into a single mRNA. Such genes are expressed as a unit and their products tend to be closely related in function. For instance, the genes in an operon often code for enzymes involved in the reactions of the same biochemical pathway.

With regard to data analysis the existence of operons necessitates some considerations: Firstly, co-expression within an operon creates strong associations in the data on the respective genes – these associations must be distinguished from associations due to regulatory links. For organisms with sequenced genomes the relative location of the corresponding DNA segments can be combined with clustering techniques to reliably identify active operons from expression data. Secondly, operons significantly reduce the dimensionality of data-driven network induction problems (regulation can be modeled via interactions between operons rather than individual genes), this enables the application of computational intelligence techniques to induce networks in spite the low number of tested conditions (typically < 100) in comparison to the protein-coding genes (e.g. ~ 4300 in *E. coli*, ~ 4200 in *B. subtilis*). Thirdly the fact that expression for genes in an operon is strongly dependent leads to undesired biases in method evaluation (e.g. for functional classification), if not properly taken into account, as expression vectors for genes from the same operon effectively duplicate each other.

3 Data Sources

3.1 Sequence and Transcriptome Data

Although there are ongoing efforts to study cells on the level of interactions within the proteome, current experimental techniques and the infrastructure supporting them are geared towards obtaining the genetic information and measuring mRNA transcript levels in samples. Several reasons can be named for this. Firstly protein identification requires comparatively new technologies and specialized equipment and expertise [3]. In contrast transcriptome-oriented techniques such the various incarnations of microarray technology, have been able to capitalize on technology, experience and manufacturing capacity of the semiconductor industry and are now widely available at comparatively low costs. A related argument is that transcriptomics have reached a higher degree of standardization in equipment, data formats and software pipelines for subsequent analysis. Finally, a large number of mRNAs can be detected on a single chip, allowing to cover the full transcriptome while reducing error sources due to biological variance and minor differences in experiment or subsequent treatment. For these reasons mRNA expression data, are often used as a proxy for indicating the presence of the corresponding protein products [22]. They are arguably the most readily available data source for the study of regulations. An analysis using microarray technology will usually reveal genes affected by a tested condition and often indicate associations between gene expression patterns. However, in bacteria it usually does not allow to identify the direction of potential regulatory interactions reliably because samples refer to populations rather than individual cells and resolution in time is limited by contraints on the number and rate for drawing samples.

To obtain expression data on a genome wide scale it is necessary to determine nucleotide sequences for the DNA of the organism. A number of sequencing technologies are available and genomes for many organism have been submitted to public databases. The sequence data is then matched to known gene sequences for the target and related species [4] that link them to specific DNA sites. In addition standard sequence analysis algorithms search for from characteristic sequences of DNA that are associated with the beginning and end of transcribed regions to identify potential genes. For the network inference problem, cultures are grown under diverse condition to cover a wide range of regulatory pathways. Samples from those cultures are collected and analyzed for their mRNA content, which is compared to a reference. By matching observed mRNA to corresponding DNA sequences the expression of sections of DNA and therefore the genes represented by that DNA can be inferred. The result is a relative quantification of the expression of genes under different conditions. On occasion the pattern of DNA expression itself reveals the presence of previously undetected features, such as new genes or small DNA fragments that affect the cell directly via their RNA transcripts rather than proteins.

3.2 Other Data Sources

For an increasing number of organisms, known genes are now annotated with information about known biological function of their products [2]. This information often provides critical clues about the role of the gene in regulatory networks, for instance, many transcription factors will bind at more than one site.

Additional information about potential regulations can be obtained my combining microarrays with chromatin immunoprecipitation technology (ChIP-on-chip). In this type of experiment selected proteins are tested for their affinity to bind to specific DNA sequences on the microarray. *In vivo* such binding events occur when regulatory proteins affect the transcription of the associated genes, but the method is known to produce a number of false positives. Nevertheless protein binding data can provide an initial directed graph of potential regulatory links to be refined when evidence from other sources becomes available.

A richer type of data can be gathered from perturbation experiments, in which the level or activity of the mRNA or protein for a targeted gene is artificially altered to observe subsequent changes in the expression of other genes. Unless the a gene is essential to the viability of the organism, knock-out experiments targeting that gene will also affect the expression of other genes that are directly or indirectly regulated by its product. Such experiments are used to reveal causal chains in regulation processes. Unfortunately the underlying techniques, which rely on mutant strains or RNA interference technology, are comparatively expensive and labor intensive, so they are usually applied on a smaller scale to validate hypothesized regulatory links supported by other pieces of evidence.

4 Determining Gene Function: Clustering and Classification

Given expression data for bacteria over several conditions the first task is to identify operons. Because expression of genes within the same operon is highly correlated, operons can be identified by clustering expression vectors for genes over several experiments. With additional information about collocation of genes and known transcription starting and termination sites the identification of operons is now considered a fairly straightforward task. In explorative analysis, expression data sets are often visualized as matrices that associate genes with rows, conditions as columns and display log-ratios of expression values in the experiment versus a control as the cell value. The application of hierarchical clustering to rearrange rows and columns in the resulting matrix according to correlation has become a common practice due to its visual appeal and early use within the in the community [7].

Assuming new genes or operons have been found however, further investigations will depend on hypotheses about their functions. This classification task is often approached by supplementing the clustering result from the exploratory step with an ad-hoc assignment based on common labels of known genes within the same cluster. The resulting strategy can be seen as an informal approximation of semi-supervised clustering techniques. Among true classification methods the k-nearest

neighbor approach has been applied successfully due to its capacity to model multi-centered classes. For the assignment to coarser functional categories based on a time series over expression data, Bayesian classifiers have recently provided valuable results, due to their ability to model associations between correlated input attributes and their explicit representation of uncertainty in category assignments (www://basysbio.org/nutrientshift, suppl. information for [6]).

5 Regulatory Network Search: Association Measures and Bayesian Network Induction

The detection of links in Gene Regulatory Networks relies on measuring associations in observed data. Unfortunately such observed associations provide only weak evidence for regulatory links. Apart from direct causal links, associations are often produced due to unavoidable systematic biases in the experiments. Moreover, they can result from indirect connections (e.g. common cause) or stochastic effects in conjunction with noisy signals and small sample size. Nevertheless the application of simple thresholds on correlation coefficients or other association measures such as mutual information, is still promoted as a network induction technique (relevance networks).

More advanced methods take the context of observed associations into account and test for conditional independence when seeking sets of causal link that potentially explain observations. In [10] the authors induce and evaluate Bayesian Networks over a selected subset of the baker's and brewer's yeasts (*Saccharomyces cerevisiae*) genome on a large compendium of microarray data. In this study two types of approaches are used to model value distributions. The first method relies on discretized expression data resulting in multinomial distributions. The other approach is parametric and models normal value distributions over the domains of all variables (Gaussian Graphical Model). The search procedure tests network topologies constructed from small sets of candidate links, which are pre-selected based on local measures of associations between the expression of genes [9], and fitting the data to each resulting topology. Causal Bayesian Networks [17] have been used with perturbation data obtained from measurements on single cells [21]. A comparison of these model types is given in [23].

More recently the DREAMS initiative has lead to systematic comparisons of inference methods for gene regulatory networks against artificial data and reference networks [15].

6 Exploring Regulatory Pathways: Frequent Pattern Mining, Enriched Ontologies and Triangular Norms

Whereas in the previous section we saw network induction as a self-contained process, biomedical research projects rarely apply network induction on the global scale. More often than not, they are centered around specific questions that seem

to offer a path to potential treatments, e.g. regulatory events in host pathogen interaction during the course of infections. In this context experiments will tend to focus on specific set of closely related conditions that draw on a limited subset of the organisms regulatory repertoire. In addition, prior knowledge from detailed low throughput studies is usually available to provide a scaffold of known regulatory interactions. In some cases specific genetic features or environmental conditions are associated with an observable effect that cannot be explained from the pre-existing knowledge about regulatory interactions.

Data from such studies is linked to phenotypcial effects (e.g. weak vs. strong immune response, phases of infection, presence of signaling molecules). In such experiments researchers aim to answer biological questions such as:

1. Which sets of genes/operons are linked to which effects/outcomes?
2. What is the (usual) role of the associated gene product in the organism?
3. How are their expression patterns linked to each other?

The first two questions identify participants in potentially new regulatory pathways and reveal gaps in the existing models of gene regulation. The third question focuses on detailed mechanism that can be tested and possibly exploited in drug development. In the following, we are assuming a typical database of microarray data as it would be generated using standard sample preparation and normalization and primary data processing procedures. Each entry in that database corresponds to a vector describing the ratios of gene expression values in a sample in which the targeted effect was triggered in relation to those measured in an associated control experiment (e.g. an uninfected sample). Again, IDA provides tools to extract relevant information from such data.

6.1 Finding Relevant Subsets of Genes and Operons

The analysis of individual microarrays focuses on genes that are over- or underexpressed in an investigated condition relative to a neutral control condition. Due to the noisy nature of such measurements, it is a common procedure to disregard genes unless the relative difference between observed expression and reference value corresponds to a factor of at least 2. A drawback of this method is that some relevant interactions may remain below the chosen threshold while biological variation in gene expression may lead to false-positive selections.

If several replicates or cases are available, statistical approaches to the detection of reoccurring gene sets supplement this selection process. Recent developments in frequent item set mining have produced algorithms suitable to support large item bases and show promise for the analysis of expression data and genome-wide-association studies [5]. Applying frequent item set mining to microarray data permits to lower such the filter criteria yet robustly identify associations between effects and associated gene sets.

6.2 Interpretation via Functional Annotations

While the identification of relevant gene sets is a necessary step in the analysis, a perspective based on the associated genes' functions provides an orientation aid for the interpretation. Gene annotation databases (such as those using the Gene Ontology [2]) record functions previously attributed to genes and may thus point out connections to other cell functions. For instance, certain bacteria that switch active metabolic pathways in the presence of preferred nutrients start to express gene sets normally related to famine stress conditions [6]. However, as these effects do not manifest themselves very strongly, they are easily overlooked in an analysis based on individual genes. An interesting point about this particular observation was, that the changes in expression of stress-associated started even before the metabolic transition took effect (as if in anticipation of a possible energy crisis) leading to a follow-up investigation of a new regulatory mechanism.)

Models for set-valued data, e.g. [19], can be used to explicitly represent imprecision w.r.t. gene function that occurs due to several roles of gene products in the organisms. They provide a function-oriented perspective to complex expressions patterns associate them with other patterns that involve different gene sets, but correspond to related functions. Because the genes involved in controlling or modulating these function are often known, this type of analysis helps define perturbation experiments that confirm the existence of regulatory paths between previously unconnected sections of the regulatory network models.

6.3 Finding Regulatory Pathways

Once the existence of a regulatory pathway between different sections of the network has been established, the causal chain leading from the original perturbation to the observed effect needs to be explored. An approach to this task proposed in [20] uses association measures and t-norms to assign a measure of plausibility to candidate paths via intermediates. The algorithm combines this approach with an efficient heuristic search strategy and on-demand evaluation of association measures to create a superposition the top k plausible pathways. Alternative output modes provide selections of candidate genes for validation by knock-out experiments. The above approach is implemented in and supported by a collection of command-line tools available from the authors website (manuscripts in preparation).

The choice of t-norms as operators for aggregating evidence for local regulatory links to assess the consistency of pathway-level hypotheses is rooted in their interpretation as extensions of logical conjunctions. Moreover the monotonicity w.r.t. both arguments is exploited in the heuristic search strategy.

6.4 Implementation in Application Software

An extensible network induction bundle (CYNI) for the network data integration, analysis and visualization software Cytoscape is currently under development. It

features a modular architecture and aims to promote applications of Intelligent Data Analysis in network biology by providing resources for method implementation. The bundle will be released with a selection of pre-implemented plug-ins that cover the whole network induction process. This offers usable network inductions pipelines to experimenters and serves as framework for extensions, modifications and systematic comparative evaluation for other method developers.

7 Current Issues and Further Reading

The methods presented in this paper focussed on structure learning tasks for detecting previously unknown links in gene regulatory networks. Such links can be validated, e.g. in knockout or RNA interference experiments. Arkin and Shaffer [1] provide a wider perspective on network induction techniques by assessing them according to their position on the descriptive/predictive dimension and their relative emphasis on associations observed in data versus mechanistic explanations.

A summary of general network induction strategies and experiments to identify cancer-related mutations via their effect on gene regulation is given in [18]. Finally, for an extended coverage of predictive models for gene expression, ranging from boolean predictions via discretized value domains, systems of ordinary differencial equations (ODEs) to extensive stochastic simulations on the level of individual molecules the reader is referred to [12]. It is remarked however, that these detailed models rely on fitting parameters to predefined network structures and that they make specific assumptions about the regulatory mechanisms. Thus their application calls for extensive prior knowledge, while parameter estimation and evaluation are dependent on rich expression, protein and/or metabolome data sets.

Although detailed quantitative models have been successfully applied for metabolic networks of biochemical pathways, their application to gene regulatory networks draws on optimistic assumptions about the mechanisms and complexity of genetic regulation in living organisms. In particular, recent studies about variability in mRNA degradation rates, the modulating effects of small RNAs, posttranslational modifications and alternative splicing undermine many simple assumptions about genetic regulation [14, 22]. This issue is particularly problematic due to the lack of extensive evaluation data sets that would allow to systematically assess the effect of relaxed assumptions for the quality of predictions. Evaluation results on synthetic data sets seemingly support those models, but the generation of such the data draws on the very assumptions challenged.

The evaluation of inference methods for network structure on the other hand, is now supported by extensive knowledge databases and expression compendia [13, 8]. These ressources enable comparisons between predicted links and archives of validated regulatory interactions. It should be noted, however, such databases are necessarily incomplete and subject to a selection bias as rare or weak effects are less likely to have been investigated before. Moreover, their use requires extensive additional processing as databases are geared towards detailed knowledge representation rather than systematic computational evaluation. To assemble a gene regulatory

network with the EcoCyc resource, for instance, relations over gene locations, proteins and protein complexes with specific regulatory functions, their binding sites, and the location of potential target genes relative to these binding sites need to be combined while considering the reading direction of the respective DNA strand during the transcription process. Implementing such an operation requires domain knowledge about transcription, regulation mechanisms and conventions in genetics in addition to general programming skills and familiarity with relational knowledge representations. This creates a barrier for integration into tools or evaluation frameworks.

8 Conclusion

With the availability and increasing reliance on automated methods for hypothesis generation and large scale network induction in biology the issue of method selection and thorough evaluation has become essential to the progress of the field. Yet the still prevalent division into modeling and experimentalist communities limits the access to critical expertise for both groups. This challenge is neither adressed by indescriminately transplanting approaches from other fields nor by the introduction of ever new postulates that are difficult to assess with currently available data. Instead it would seem prudent to form closer collaborations between the Network Biology and Intelligent Data Analysis communities to improve knowledge about the properties of biological datasets, identify stengths and weaknesses of existing models, adapt methods to current tasks and foster a culture of systematic evaluation and documentation of models and the conditions, under which they can be applied.

Acknowledgements. The author participates in the development of Cytoscape. He was supported by the NRNB, U.S. National Institutes of Health, National Center for Research Resources (grant numbers P41 RR031228 and GM103504).

References

[1] Arkin, A.P., Schaffer, D.V.: Network news: Innovations in 21st century systems biology. Cell 144(6), 844–849 (2011)

[2] Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M., Sherlock, G.: Gene Ontology: tool for the unification of biology. Nat. Genet. 25, 25–29 (2000)

[3] Becher, D., Büttner, K., Moche, M., Heßling, B., Hecker, M.: From the genome sequence to the protein inventory of bacillus subtilis. Proteomics 11(15), 2971–2980 (2011)

[4] Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Sayers, E.W.: Genbank. Nucleic. Acids Res. 38(suppl. 1), D46–D51 (2010)

[5] Borgelt, C., Yang, X., Nogales-Cadenas, R., Carmona-Saez, P., Pascual-Montano, A.: Finding closed frequent item sets by intersecting transactions. In: Proc. 14th Int. Conf. on Extending Database Technology, EDBT 2011, Uppsala, Sweden, pp. 367–376. ACM Press, New York (2011)

[6] Buescher, J.M., Liebermeister, W., Jules, M., Uhr, M., Muntel, J., Botella, E., Hessling, B., Kleijn, R.J., Le Chat, L., Lecointe, F., Mäder, U., Nicolas, P., Piersma, S., Rügheimer, F., Becher, D., Bessieres, P., Bidnenko, E., Denham, E.L., Dervyn, E., Devine, K.M., Doherty, G., Drulhe, S., Felicori, L., Fogg, M.J., Goelzer, A., Hansen, A., Harwood, C.R., Hecker, M., Hubner, S., Hultschig, C., Jarmer, H., Klipp, E., Leduc, A., Lewis, P., Molina, F., Noirot, P., Peres, S., Pigeonneau, N., Pohl, S., Rasmussen, S., Rinn, B., Schaffer, M., Schnidder, J., Schwikowski, B., van Dijl, J.M., Veiga, P., Walsh, S., Wilkinson, A.J., Stelling, J., Aymerich, S., Sauer, U.: Global network reorganization during dynamic adaptation of Bacillus subtilis metabolism. Science 335(6072), 1099–1103 (2012)

[7] Eisen, M.B., Spellman, P.T., Brown, P.O., Botstein, D.: Cluster analysis and display of genome-wide expression patterns. Proc. Natl. Acad. Sci. USA 95(25), 14,863–14,868 (1998)

[8] Faith, J.J., Driscoll, M.E., Fusaro, V.A., Cosgrove, E.J., Hayete, B., Juhn, F.S., Schneider, S.J., Gardner, T.S.: Many microbe microarrays database: uniformly normalized affymetrix compendia with structured experimental metadata. Nucleic. Acids Res. 36(suppl. 1), D866–D870 (2008)

[9] Friedman, N., Koller, D.: Being bayesian about network structure. In: Proc. Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI 2000), pp. 201–210 (2000)

[10] Friedman, N., Linial, M., Nachman, I., Pe'er, D.: Using bayesian networks to analyze expression data. J. Comput. Biol. 7(3/4), 601–620 (2000)

[11] Jacob, F., Perrin, D., Sanchez, C., Monod, J.: L'operon: groupe de génes á expression coordoneé par un operateur. Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences 250(6), 1727–1729 (1960) (in French)

[12] Karlebach, G., Shamir, R.: Modelling and analysis of gene regulatory networks. Nat. Rev. Mol. Cell Biol. 9(10), 770–780 (2008)

[13] Keseler, I.M., Collado-Vides, J., Santos-Zavaleta, A., Peralta-Gil, M., Gama-Castro, S., Muñiz-Rascado, L., Bonavides-Martinez, C., Paley, S., Krummenacker, M., Altman, T., Kaipa, P., Spaulding, A., Pacheco, J., Latendresse, M., Fulcher, C., Sarke, M., Shearer, A.G., Mackie, A., Paulsen, I., Gunsalus, R.P., Karp, P.D.: Ecocyc: a comprehensive database of *Escherichia coli* biology. Nucleic. Acids Res. 39(Database issue), D583–D590 (2011)

[14] Licatalosi, D.D., Darnell, R.B.: Rna processing and its regulation: global insights into biological networks. Nat. Rev. Genet. 11, 75–87 (2010)

[15] Marbach, D., Prill, R.J., Schaffter, T., Mattiussi, C., Floreano, D., Stolovitzky, G.: Revealing strengths and weaknesses of methods for gene network inference. Proc. Natl. Acad. Sci. 107(14), 6286–6291 (2010)

[16] Nicolas, P., Mäder, U., Dervyn, E., Rochat, T., Leduc, A., Pigeonneau, N., Bidnenko, E., Marchadier, E., Hoebeke, M., Aymerich, S., Becher, D., Biscicchia, P., Botella, E., Delumeau, O., Doherty, G., Denham, E.L., Fogg, M.J., Fromion, V., Goelzer, A., Hansen, A., Härtig, E., Harwood, C.R., Homuth, G., Jarmer, H., Jules, M., Klipp, E., Le Chat, L., Lecointe, F., Lewis, P., Liebermeister, W., March, A., Mars, R.A.T., Nannapaneni, P., Noone, D., Pohl, S., Rinn, B., Rügheimer, F., Sappa, P.K., Samson, F., Schaffer, M., Schwikowski, B., Steil, L., Stülke, J., Wiegert, T., Devine, K.M., Wilkinson, A.J., van Dijl, J.M., Hecker, M., Völker, U., Bessières, P., Noirot, P.: Condition-dependent transcriptome reveals high-level regulatory architecture in Bacillus subtilis. Science 335(6072), 1103–1106 (2012)

[17] Pearl, J.: Causality: Models, Reasoning, and Inference. Cambridge University Press, New York (2000)

[18] Pe'er, D., Hacohen, N.: Principles and strategies for developing network models in cancer. Cell 144(6), 864–873 (2011)
[19] Rügheimer, F.: Using Enriched Ontology Structure for Improving Statistical Models of Gene Annotation Sets. In: Hüllermeier, E., Kruse, R., Hoffmann, F. (eds.) IPMU 2010. CCIS, vol. 81, pp. 55–64. Springer, Heidelberg (2010)
[20] Rügheimer, F., Anand, A., Schwikowski, B.: A software architecture for *De Novo* induction of regulatory networks from expression data. In: Proceedings of the Journées Ouvertes Biologie Informatique Mathématiques (JOBIM) 2011, Paris, pp. 295–296 (2011)
[21] Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D.A., Nolan, G.P.: Causal protein-signaling networks derived from multiparameter single-cell data. Science 308(5721), 523–529 (2005)
[22] Vogel, C., Marcotte, E.M.: Insights into the regulation of protein abundance from proteomic and transcriptomic analyses. Nat. Rev. Genet. 13(4), 227–232 (2012)
[23] Werhli, A.V., Grzegorczyk, M., Husmeier, D.: Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical gaussian models and bayesian networks. Bioinformatics 22(20), 2523–2531 (2006)

From Spatial Data Mining in Precision Agriculture to Environmental Data Mining

Georg Ruß

Abstract. In the first part of this article, the main results from applying data mining methods and algorithms to spatial precision agriculture data sets will be outlined. In particular, the task of yield prediction will be handled as a spatial regression problem. To account for the spatial nature of the data sets, a few modeling pitfalls resulting from spatial autocorrelation will be tackled. Based on a cross-validation approach, the yield prediction setting will be used to determine spatial variable importance. Another task called management zone delineation will be briefly outlined. A novel hierarchical spatially constrained clustering algorithm will be presented which aims to provide a tradeoff between spatial contiguity of the resulting clusters and cluster similarity. These two tasks are a summary of [26]. In the second part of this article, the emerging field of *environmental data mining* will be briefly laid out.

1 Introduction

While the (spatial) data sets around us grow rapidly, the tools and algorithms to match those data sets are struggling to keep up. While geographical information systems and location-based services are rapidly expanding, the agricultural sector is currently experiencing an influx of information technology, mostly based on the global positioning system and technological advances in sensors and data aggregation. However, even *precision agriculture* is still in its infancy and requires novel data mining tools and algorithms adapted for the special spatial data sets.

Agricultural companies nowadays harvests not only crops but also growing amounts of data. These data are site specific – which is essentially why the combination of GPS, agriculture and data has been termed *site-specific crop management*. A large amount of information about the soil and crop properties enabling a higher operational efficiency is often contained in these spatial data sets – appropriate

Georg Ruß
TecData AG, 9240 Uzwil, Switzerland
e-mail: georg.russ@buhlergroup.com

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 263–273.
springerlink.com

techniques should therefore be applied to find this information. This is a rather common problem for which the term *data mining* has been coined. Data mining techniques aim at finding those patterns in the data that are both valuable and interesting for crop management. This article primarily summarizes the author's two main lines of research. Furthermore, it extends the work on *data mining in precision agriculture* towards a broader scope on *environmental data mining*. The first two parts shortly recapitulate existing work based mainly on [27] and [25].

2 Data Description

The data available in this work were collected during the growing season of 2007 on three sites south of Köthen, Germany. The data for the sites, called *F440*, *F611* and *F631*, respectively, were interpolated using kriging [30] to a grid with 10 by 10 meters grid cell sizes. Each grid cell represents a record with all available information. The fields grew winter wheat. Nitrogen fertilizer (N) was applied three times during the growing season. Overall, for each field there are six input attributes, accompanied by the respective current year's yield (2007) as the target attribute. In total, there are 6446 (F440), 4970 (F611) and 7875 records (F631).

Yield is measured in metric tons per hectare ($\frac{t}{ha}$), along the harvesting lanes (spaced 8 m apart), roughly every ten meters. Apparent electrical soil conductivity (EC25) as a measure for a number of soil properties is acquired. Satellite or aerial image processing provides a measure of vegetation called the red edge inflection point (REIP) value, at two points into the growing season (REIP32, REIP49), according to the growing stage defined in [17]. The REIP value may also be used directly for guiding fertilizations [10]. A simplified assumption is that a higher REIP value means more vegetation. Three nitrogen fertilizer dressings are applied (N1, N2, N3, in $\frac{kg}{ha}$). In the available data, due to the fields being experimental agriculture sites, the nitrogen dressings were not temporally autocorrelated. However, this phenomenon may be considered in production sites. EC, REIP and N are measured in 10-m-intervals along the lanes which are spaced 24 meters apart.

3 Spatial Cross-Validation and Regression

According to [9], *spatial autocorrelation* is the correlation among values of a single variable strictly attributable to the proximity of those values in geographic space, introducing a deviation from the *independent observations* assumption of classical statistics. Given a spatial data set, spatial autocorrelation can be determined using Moran's I ([18]) or semivariograms. For the data sets used in this article, each of the attributes exhibits spatial autocorrelation. In practice, it is usually also known from the data origin whether spatial autocorrelation exists. For further information it is referred to, e.g., [3].

In previous articles using the above data, such as [28, 24], the main focus was on finding a suitable regression model to predict the current year's yield sufficiently well. However, the used regression models, such as neural networks [28, 29] or

support vector regression [24], among others, generally assume statistical independence of the data records. However, with the given geo-tagged data records at hand, this is clearly not the case, due to (natural) spatial autocorrelation. Therefore, the spatial relationships between data records have to be taken into account.

Due to the shortcomings in classical regression and cross-validation learning approaches when using them on spatial data, this section will present a novel regression model for data sets which exhibit spatial autocorrelation. In non-spatial regression models, data records which appear in the training set are not supposed to appear in the test set during a cross-validation learning setup. Classical sampling methods do not take spatial neighborhoods of data records into account. Therefore, the above assumption may be rendered invalid when using non-spatial models on spatial data. This inevitably leads to overfitting and underestimates the true prediction error of the regression model (compare [1, 2] for similar observations in a classification context). Therefore, the main issue is to avoid having neighboring or the same samples in training and testing data subsets during a cross-validation learning approach. The basic idea therefore is to apply changes to the resampling method and keep the regression modeling techniques as-is. The resulting procedure can be seen as spatial cross-validation technique.

Traditionally, cross-validation for regression randomly subdivides a given data set into two or three parts: a training set, (optionally a validation set) and a test set. A 10- to 20-fold cross-validation is usually considered appropriate to remove bias [14]. The regression model is trained on the training set until the prediction error on the validation set starts to rise. Once this happens, the training process is stopped and the error on the test set is reported for this fold. This procedure is repeated r times to remove a possible sampling bias. In our case, r has been empirically determined as 100. Instead of sampling randomly from the data set to generate the training and test sets, a clustering step is inserted. A simple k-Means clustering on the data records' x/y-coordinates yields a spatial tessellation of the site under study. The sub-areas of the site are roughly the same size on typical sites. Once the tessellation exists, the cross-validation samples randomly from the sub-areas rather than from the whole data set. The regression is then performed on the data records within the sampled sub-areas. Since k-Means is sensitive to initialization, the clustering is repeated r times.

The spatial clustering procedure may thus be considered as a broader definition of the standard cross-validation setup. This can be seen as follows: when refining the clustering further, the spatial zones on the field become smaller. The border case is reached when the field is subdivided into as many clusters as there are data records, i.e. each data record describes its own cluster. In this special case, the advantages of spatial clustering are lost since no spatial neighborhoods are taken into account in this approach. Therefore, the number of clusters should be seen as a tradeoff between predictive precision and statistical validity of the model. The parameter k for the size of the tessellation has to be determined heuristically.

In previous work ([24, 28]), numerous regression modeling techniques have been compared on similar data sets to determine which of those modeling techniques works best. Support vector regression has been determined as the best modeling

technique. It has furthermore recently been shown to work rather successfully in spatial classification tasks, albeit without spatial cross-validation, as in [21]. Hence, in this work support vector regression will serve as a benchmark technique against which further models will have to compete. The techniques and the respective R packages used here are *support vector regression* (e1071), *regression trees* (rpart), *random forests* (randomForest), *bagging* with trees (ipred). The models' performance is measured via the root mean squared error (RMSE) between actual value and predicted value.

The results in Table 1 confirm that the spatial autocorrelation inherent in the data set leads classical, non-spatial regression modeling setups to a substantial underestimation of the prediction error. This outcome is consistent throughout the results, regardless of the used technique and regardless of the parameters. Furthermore, it can be seen that Random Forests seem to yield better performance in terms of lower prediction error, regardless of the setup used. Moreover, the spatial setup can be easily set to emulate the non-spatial setup: set k to be the number of data records in the data set. Therefore the larger the parameter k is set, the smaller the difference between the spatial and the non-spatial setup should be. This assumption also holds true for almost all of the obtained results.

Table 1 Results of running different setups on the data sets F440 and F611; comparison of spatial vs. non-spatial treatment of data sets; root mean squared error in t/ha is shown, averaged over clusters/folds; k is either the number of clusters in the spatial setup or the number of folds in the non-spatial setup. The average yield is around 8-10 t/ha.

		F440		F611	
	k	spatial	non-spatial	spatial	non-spatial
Support Vector Regression	10	1.06	0.54	0.73	0.40
	20	1.00	0.54	0.71	0.40
	50	0.91	0.53	0.67	0.38
Regression Tree	10	1.09	0.56	0.69	0.40
	20	0.99	0.56	0.68	0.42
	50	0.91	0.55	0.66	0.40
Random Forest	10	0.99	0.50	0.65	0.41
	20	0.92	0.50	0.64	0.41
	50	0.85	0.48	0.63	0.39
Bagging	10	1.09	0.59	0.66	0.42
	20	1.01	0.59	0.66	0.42
	50	0.94	0.58	0.65	0.41

4 Management Zone Delineation with HACC-Spatial

The second task in precision agriculture which is summarized in this article is *management zone delineation*. In brief, it aims to generate a subdivision of the site under study into homogeneous zones which are, to a certain degree, contiguous in space. Further details can be acquired from [25]. From a data mining point of view, this task

is essentially a clustering challenge where a specific constraint (spatial contiguity) must be taken into account accordingly.

The underlying idea of HACC-SPATIAL is to adapt hierarchical agglomerative clustering (HAC) for spatial data sets. In HAC, the clustering of arbitrary objects starts with each object in a single cluster. Consecutively, two clusters are merged into one new cluster: the decision which clusters to merge is often done based on cluster similarity or distance, using an appropriate distance measure. Furthermore, constraints have been introduced into HAC, leading to hierarchical agglomerative constrained clustering (HACC): the decision which clusters to merge is not only done based on the similarity, but also according to constraints which can have two types. The first is a *must-link* constraint, which means that two clusters belong into one cluster. The second is of the opposite *cannot-link* constraint, which determines that two clusters must not be merged. The idea of HACC-SPATIAL is now to use a *cannot-link* constraint to enforce spatial contiguity of the resulting clusters. Furthermore, in the beginning of the clustering the algorithm strictly enforces spatial contiguity due to the constraint, while the constraint may be relaxed after a certain threshold between adjacent and non-adjacent clusters is reached.

The cluster distance is determined in feature space, while the constraint ensures spatial contiguity in geographic space. For lower-dimensional feature spaces, Euclidean distance is used, while for higher dimensions, due to the curse of dimensionality, the Cosine distance may be used. The details of HACC-SPATIAL can be obtained from [25].

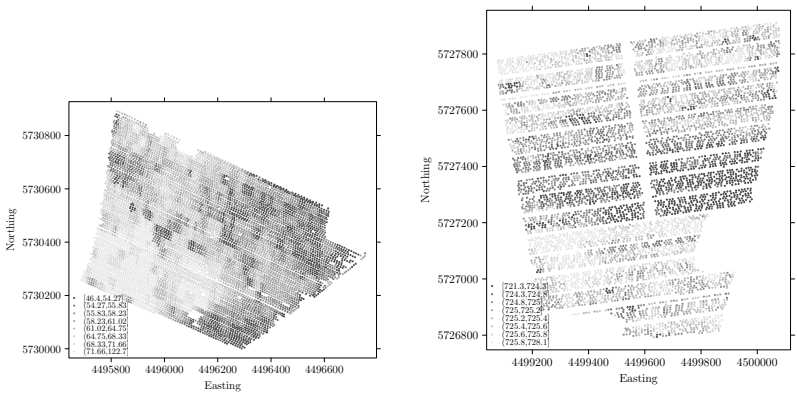
To exploit spatial autocorrelation (which is typically present in precision agriculture data sets) and reduce the computational burden, HACC-SPATIAL does not start directly with each data object in a single cluster. Instead, it can safely be assumed that a few spatially adjacent data objects are similar enough to be put into an initial cluster. To achieve this initial clustering, a round of k-Means clustering is applied initially to the spatial coordinates of the data objects. Depending on the heterogeneity of the site, the number of initial clusters in the tessellation which is generated by k-Means should be set in a range of around 100 to N, where N is the number of data objects to be clustered.

4.1 Results on Different Precision Agriculture Data Sets

The two variables from two actual sites which HACC-SPATIAL will be applied to are depicted in Figure 1. While the REIP value alone has no practical use in this zone delineation task, it certainly is of major importance in other YIELD-related tasks. The experiments are designed such that the algorithm's results can be easily visually compared with the actual variable under study. Practically, zone delineation is often done using the EC variable.

4.1.1 F631

A result demonstrating the different settings of the contiguity parameter is presented in Figure 2, where the variable EC25 of the F631 field is used for management zone



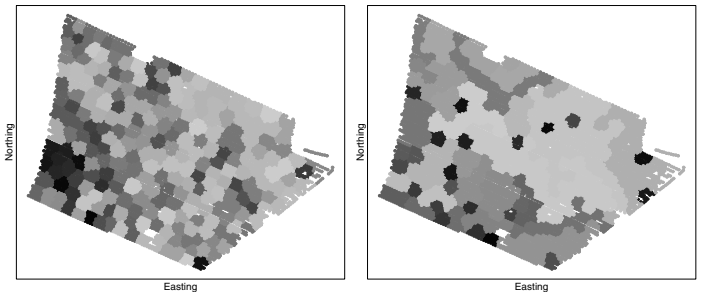
(a) F631: EC25 (b) F440: REIP32

Fig. 1 F631: EC25, F440: REIP32. The spatial distribution of the EC25 and the REIP32 variables clearly exhibits spatial autocorrelation. In the bottom figure, strips of data are missing, while the underlying distribution can still be identified by a human. An appropriate clustering algorithm has to be developed to generate management zones from this variable.

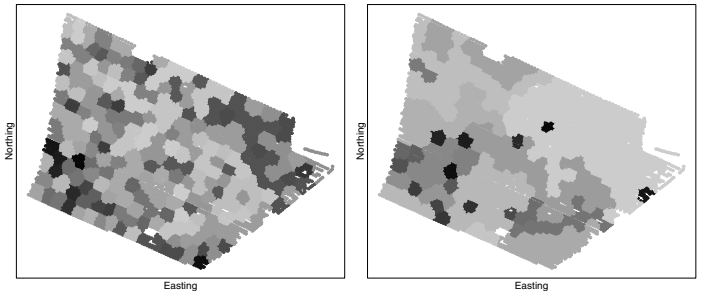
delineation. The field is initially tessellated into 250 clusters and the clustering is run with low and high contiguity settings to compare the results. Clustering with low spatial contiguity yields mostly non-contiguous clusters (as expected) until spatially contiguous clusters start emerging towards the very end of the clustering (Figure 2e). On the other hand, clustering with high spatial contiguity starts showing emergent clusters after around 200 merging steps (Figure 2b) and subsequent clusters clearly correspond to the actual variable value (Figure 1a). The clusters are not limited to convex shapes and account for the irregular shape of the field (missing data, irregular borders, "holes"). If the clustering in Figure 2f is deemed too coarse, the hierarchically structured clustering easily allows for subdividing single clusters by traversing the dendrogram.

4.1.2 F440

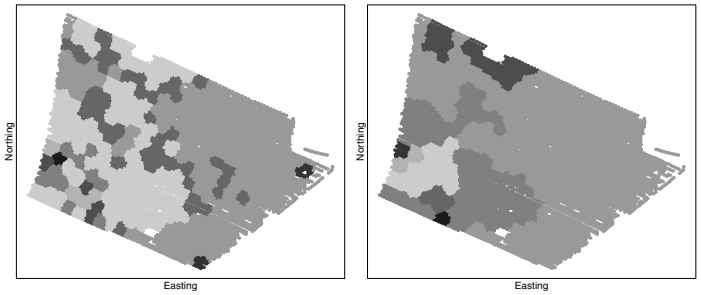
In the preceding sections, the main purpose was to show the effect of enforcing or neglecting spatial contiguity throughout the clustering. This was achieved by setting the contiguity ratio threshold accordingly. A direct comparison of the results of HACC-SPATIAL when applied to the same input data is provided here. Figure 3 shows the REIP32 variable on the F440 field, clustered by HACC-SPATIAL, showing the stage at which 15 clusters are left. While Figure 3a shows almost no visible spatial contiguity, this changes gradually towards Figure 3d where the clusters are clearly spatially contiguous.



(a) F631, EC25, 50 clusters (b) F631, EC25, 50 clusters

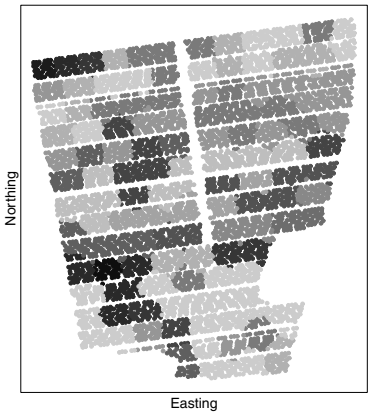


(c) F631, EC25, 30 clusters (d) F631, EC25, 30 clusters

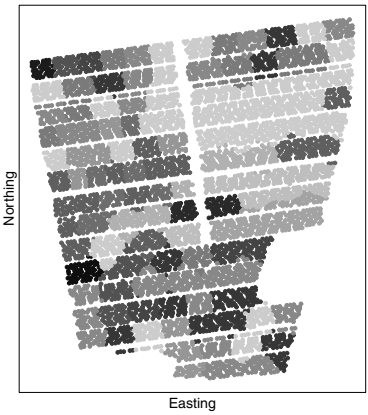


(e) F631, EC25, 8 clusters (f) F631, EC25, 8 clusters

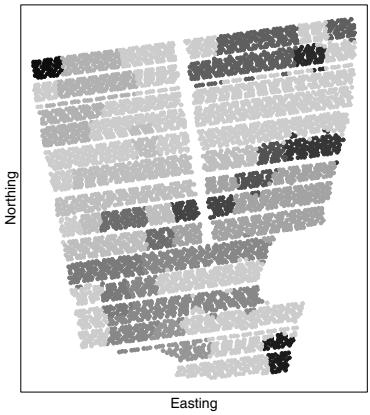
Fig. 2 HACC-SPATIAL on F631, using the variable EC25 (cp. Figure 1a on Page 268), starting with 250 clusters. Clustering with low (left figures) and high (right figures) spatial contiguity shows considerable differences in the spatial structure of the resulting clusters. At low spatial contiguity the algorithm starts producing visible spatially contiguous clusters only towards the end of the clustering (e), while spatially contiguous clusters start emerging much earlier when clustering with high spatial contiguity (b).



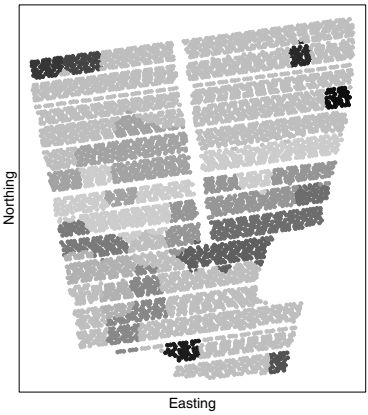
(a) F440, REIP32, 15 clusters



(b) F440, REIP32, 15 clusters



(c) F440, REIP32, 15 clusters



(d) F440, REIP32, 15 clusters

Fig. 3 HACC-SPATIAL on F440, 120 initial clusters, using the REIP32 variable and demonstrating the effect of different spatial contiguity settings. While (a) shows spatially rather scattered clusters, the change in the designed contiguity ratio threshold varies the spatial contiguity of the clusters until spatial contiguity is strictly enforced in Figure (d).

4.2 Clustering Summary

Based on both the practical and the theoretical need for an efficient and understandable algorithm for management zone delineation in precision agriculture, a novel algorithm HACC-SPATIAL has been devised. It is able to exploit spatial autocorrelation in the precision agriculture data and successfully extends hierarchical agglomerative constrained clustering towards spatial data sets. An algorithmic description and results on one-dimensional spatial data sets have been presented. The main parameter *contiguity threshold* has been experimentally validated and shown to be successful in three practical data sets.

5 Environmental Data Mining

The original definition of data mining within the process of knowledge discovery in databases by [6] described it as "the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data". The data collected in environmental sciences such as ecology, geology, remote sensing and agriculture are by their very nature spatial and/or temporal which are important additional properties when it comes to data mining. Hence, it is proposed to extend the above definition towards environmental data mining as follows:

> *Environmental Data Mining* is the nontrivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in *spatial* and *temporal* data *from environmental sciences*.

Many of the developed techniques in data mining, though not particularly adapted for the specifics of environmental data sets, are rather flexible. They can often be tailored to fit environmental data, such as the regression and clustering problems presented in this article. Introductions to this increasingly active field can be found in [8], [13], [12] and [7].

Given the classicals tasks of classification and regression, especially the work in ecology has started around the year 2000, ranging from neural networks [15] over bayesian statistics and belief networks [16] to bagging and random forests [22],[1], [2]. The related task of clustering in environmental data sets has a history that dates back to 1990 [5], with numerous further applications of fuzzy clustering, such as in agriculture [19] and remote sensing [20]. A third frequent task in classical data mining is association analysis, which has also been introduced into ecology [31], remote sensing [11] and agriculture [4], among others.

With those prerequisites, the term *environmental data mining* encompasses most of the existing work under a common umbrella term, while distinctively combining the fields of environmental sciences and data mining.

Acknowledgements. The implementation is carried out in R [23]. The R scripts are available on request from the author of this article. The data in this work have been acquired on the experimental farm Görzig in the federal state Sachsen-Anhalt, in Germany. The data were obtained from Martin Schneider and Peter Wagner from Martin-Luther-Universität Halle-Wittenberg, Germany, Lehrstuhl für landwirtschaftliche Betriebslehre.

References

[1] Brenning, A.: Spatial prediction models for landslide hazards: review, comparison and evaluation. Natural Hazards and Earth System Science 5(6), 853–862 (2005)
[2] Brenning, A., Lausen, B.: Estimating error rates in the classification of paired organs. Statistics in Medicine 27(22), 4515–4531 (2008)
[3] Cressie, N.A.C.: Statistics for Spatial Data. Wiley, New York (1993)
[4] El-Beltagy, S.R., Rafea, A., Mabrouk, S.: Agrimine: A tool for mining agricultural problems and their solutions. In: Proc. of Int. Computer Engineering Conference (ICENCO), ICENCO, pp. 81–85 (2010)

[5] Equihua, M.: Fuzzy clustering of ecological data. Journal of Ecology 78(2), 519–534 (1990)
[6] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: The kdd process for extracting useful knowledge from volumes of data. Commun. ACM 39, 27–34 (1996)
[7] Fielding, A.: An introduction to machine learning methods. In: Fielding, A. (ed.) Machine Learning Methods for Ecological Applications, pp. 1–35. Kluwer Academic Publishers, Dordrecht (1999)
[8] Gibert, K., Spate, J., Snchez-Marr, M., Athanasiadis, I.N., Comas, J.: Data mining for environmental systems. In: Jakeman, A.J., Voinov, A.A., Rizzoli, A.E., Chen, S. (eds.) Environmental Modelling, Software and Decision Support, Developments in Integrated Environmental Assessment, vol. 3, pp. 205–228. Elsevier (2008)
[9] Griffith, D.A.: Spatial Autocorrelation and Spatial Filtering. Advances in Spatial Science. Springer, New York (2003)
[10] Heege, H., Reusch, S., Thiessen, E.: Prospects and results for optical systems for site-specific on-the-go control of nitrogen-top-dressing in germany. Precision Agriculture 9(3), 115–131 (2008)
[11] Iisaka, J., Sakurai-Amano, T.: Spatial association analysis for radar image interpretation. In: International Geoscience and Remote Sensing Symposium, IGARSS, pp. 1200–1203 (1993)
[12] Kanevski, M. (ed.): Advanced Mapping of Environmental Data: Geostatistics, Machine Learning and Bayesian Maximum Entropy. ISTE, London (2010)
[13] Kanevski, M., Parkin, R., Pozdnukhov, A., Timonin, V., Maignan, M., Demyanov, V., Canu, S.: Environmental data mining and modeling based on machine learning algorithms and geostatistics. Environmental Modelling and Software 19(9), 845–855 (2004)
[14] Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of International Joint Conference on Artificial Intelligence (1995)
[15] Lek, S., Guegan, J.: Application of artificial neural networks in ecological modelling. Ecological Modelling 120(2-3) (1999)
[16] Marcot, B.G., Holthausen, R.S., Raphael, M.G., Rowland, M.M., Wisdom, M.J.: Using bayesian belief networks to evaluate fish and wildlife population viability under land management alternatives from an environmental impact statement. Forest Ecology and Management 153, 29–42 (2001)
[17] Meier, U.: Entwicklungsstadien mono- und dikotyler Pflanzen. In: Biologische Bundesanstalt für Land- und Forstwirtschaft, Braunschweig, Germany (2001)
[18] Moran, P.A.P.: Notes on continuous stochastic phenomena. Biometrika 37, 17–33 (1950)
[19] Papajorgji, P.J., Pardalos, P.M. (eds.): Advances in Modeling Agricultural Systems. Springer Optimization and Its Applications, vol 25. Springer (2009)
[20] Petcu, D., Zaharie, D., Panica, S., Hussein, A.S., Sayed, A., El-Shishiny, H.: Fuzzy clustering of large satellite images using high performance computing. In: Proc. of SPIE. SPIE, vol. 8183 (2011)
[21] Pozdnoukhov, A., Foresti, L., Kanevski, M.: Data-driven topo-climatic mapping with machine learning methods. Natural Hazards 50(3), 497–518 (2009)
[22] Prasad, A.M., Iverson, L.R., Liaw, A.: Newer classification and regression tree techniques: Bagging and random forests for ecological prediction. Ecosystems 9, 181–199 (2006)

[23] R Development Core Team R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2009), http://www.R-project.org, ISBN 3-900051-07-0
[24] Ruß, G.: Data Mining of Agricultural Yield Data: A Comparison of Regression Models. In: Perner, P. (ed.) ICDM 2009. LNCS (LNAI), vol. 5633, pp. 24–37. Springer, Heidelberg (2009)
[25] Ruß, G.: Hacc-spatial: Hierarchical agglomerative spatially constrained clustering. In: Bichindaritz, I., Perner, P., Ruß, G. (eds.) 11th ICDM Conference, New York, USA, Workshop Proceedings. IBaI Publishing, Leipzig (2011)
[26] Ruß, G.: Spatial Data Mining in Precision Agriculture. PhD thesis, Otto-von-Guericke-Universität Magdeburg (2012)
[27] Ruß, G., Brenning, A.: Data Mining in Precision Agriculture: Management of Spatial Information. In: Hüllermeier, E., Kruse, R., Hoffmann, F. (eds.) IPMU 2010. LNCS, vol. 6178, pp. 350–359. Springer, Heidelberg (2010)
[28] Ruß, G., Kruse, R., Schneider, M., Wagner, P.: Estimation of neural network parameters for wheat yield prediction. In: Artificial Intelligence in Theory and Practice II. IFIP, vol. 276, pp. 109–118. Springer, Boston (2008)
[29] Ruß, G., Kruse, R., Schneider, M., Wagner, P.: Optimizing wheat yield prediction using different topologies of neural networks. In: Verdegay, J.L., Ojeda-Aciego, M., Magdalena, L. (eds.) Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2008), pp. 576–582. University of Málaga (2008)
[30] Stein, M.L.: Interpolation of Spatial Data: Some Theory for Kriging. Springer Series in Statistics. Springer (1999)
[31] Su, F., Zhou, C., Lyne, V., Du, Y., Shi, W.: A data-mining approach to determine the spatio-temporal relationship between environmental factors and fish distribution. Ecological Modelling 174, 421–431 (2004)

Real-Time Data Mining with In-Memory Database Technology

Matthias Steinbrecher and Joos-Hendrik Boese

Abstract. In the past, classical databases basically came in two flavors: optimized for write access (OLTP, online transaction processing) or read access (OLAP, online analytic processing). Typical data mining tasks, however, involve preprocessing, feature extraction, model training and cross-validation which cannot fully be categorized as either flavor. SAP's in-memory database HANA stores all data cache-aware in main memory, allowing for rapid transactional and analytical access. While in common three-tier architectures (database, application server, client), computationally intensive applications run at the application server layer and data is loaded into the main memory of application servers, enterprise applications developed for or moved to HANA are more tightly integrated with the database. The main principle of application development for HANA is to execute data-intensive computations in the database close to the raw data in order to prevent expensive data movement. This shift in application design poses new challenges to the application developer: in order to utilize HANA efficiently, he has to think differently about how to design his application.We'll address these challenges, discuss a real-world data analysis scenario and present some open questions in this area.

1 SAP In-Memory Database

The SAP HANA database management system [2] provides the data management infrastructure for current and future SAP enterprise applications. The technical architecture and design of HANA was driven by the significant change of requirements on data management in modern enterprise applications, as well as recent developments in hardware architectures:

Over the last decades, enterprise applications could be classified as either OLAP or OLTP centric. Today, in most cases, this distinction cannot be made anymore,

Matthias Steinbrecher · Joos Hendrik-Boese
SAP AG, 10178 Berlin
e-mail: {matthias.steinbrecher,joos-hendrik.boese}@sap.com

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 275–284.
springerlink.com

as more and more analytical applications require the immediate availability of operational data to support accurate decision making, while transactional applications also tend to touch large amounts of data, e.g., to calculate aggregates like sums of already delivered orders per client [6]. In addition, modern enterprise applications demand for non-standard features such as planning, optimization or predictive analysis functionality, which sometimes rely on non-relational data models, such as graphs or semi-structured data. As speed is a key factor for business success, response times of query execution for decision making must be in sub-second to seconds. This requires all data to be kept constantly online for fast querying and analytics. Therefore, HANA keeps the primary copy of its data in-memory to provide up-to-date data for fast ad-hoc processing in-memory at any time.

To provide a solution for the heterogeneous needs of complex enterprise applications, HANA embeds multiple storage and query engines supporting different domain-specific languages and targeting different data models:

1. To support the standard SQL features for enterprise applications, HANA provides a relational storage and query engine that allows access to relational data via SQL. The relational store can physically organize data in a column- or row-oriented fashion, depending on expected data access patterns. Organizing data along columns bears the advantage of high compression rates and cache-efficient processing of aggregation and scan operations [5]. Thus, relations that are used for data-intensive operations are stored in a column-oriented manner. In fact, scan operations on compressed columns are so fast that in the majority of cases there is no need to create and maintain indexes, which reduces the memory required to manage the data.
2. Since valuable insights can be gained from enhancing structured information with unstructured or semi-structured data, HANA embeds a text engine that supports text indexing and common text search features, such as fuzzy or phrase search. Such data can then be "joined" to relations of the relational store.
3. The graph engine provides access to graph structures, as commonly required in SAP's planning and supply chain applications. Domain-specific graph languages are supported to query and manipulate stored graph data.

To access and process the information stored by the different storage engines, domain-specific languages and extensions to SQL are provided. Besides SQL, supported languages at the moment are: (i) MDX for multi-dimensional expressions, (ii) proprietary languages for planning applications, and (iii) various extensions to SQL, e.g., for text search. Functionality that cannot be expressed in SQL or one of the domain-specific languages can be implemented using a procedural extension to SQL called SQL Script. SQL Script is a Real-time Data Mining with In-Memory Database Technology flexible programming language with imperative elements such as loops and conditionals allowing to define the control flow of applications. SQL Script can be coupled with standard (declarative) SQL and operators defined by HANA's domain-specific languages. Functionality and business logic that is frequently used in SAP's enterprise applications, such as currency conversion, is implemented in a function library natively in the database kernel. These

functions can be programmed with a maximum degree of parallelism, since the library developer is in full control and can implement parallel execution on a lower level. Therefore, these operators are much more tightly coupled with the database kernel in contrast to classical stored procedures implemented in SQL Script. These native functions can then be called in procedural SQL Script code. Stored procedures and domain-specific languages are translated into an internal query processing structure called the "Calculation Model" [3]. Calculation models may contain native operators implemented by different query engines or operators defined in the native function libraries. By integrating support for multiple data models and languages, HANA establishes a holistic data management platform for SAP's enterprise applications that allows to speed up existing applications, and enables the development of completely new types of applications.

To leverage the performance of this new database layer, applications have to be revised to push application logic down to the data, i.e., from the application layer into the database. For such analytical applications the database server will be configured not only to serve data input and output tasks but also to have enough computational capabilities to cater for the above-mentioned analysis tasks.

2 Developing the Next Generation of Business Applications

The predominant paradigm for HANA development is to "bring the algorithms to the data rather than bring the data to the algorithms". This essentially means to push application logic down to the database which contrasts the classical three-tier pattern. In the classical architecture the database was considered as the computational bottleneck and the middle-tier was responsible for processing compute-intensive operations. The rationale behind this design pattern was that the application server layer could be easily scaled-out by adding additional machines, while the database management systems could only be scaled-up, which was uneconomical. Thus, the database server had to be protected from compute-intensive operations. With the ever increasing computing power and main memory sizes of modern hardware architectures, scaling-up the database server is technically and economically feasible today as proved by the HANA appliance. It is actually nowadays that memory prices have declined enough to afford servers that are capable of storing the entire operational data of a company in-memory.

Having a database management system that integrates heterogeneous domain-specific features in addition to significant computing power, changes the way SAP's enterprise applications can be designed. To make use of the full potential of SAP's in-memory computing engine, the developer has to decide about what parts of his application are data-intensive. This part can then be pushed down into the database, where it is executed close to the primary data structures of the storage engines. This eliminates data movement from the database into application servers, which, in fact, is the main bottleneck for data-intensive applications in traditional three-tier architectures today. Identifying data-intensive operations is sometimes trivial and sometimes tricky. Application developers could for example be guided by best

practices, design patterns, tools for code analysis, profiling, etc. Another approach is to decide at execution time, whether for a given setting, either data is moved from the database to the application server or whether code should be shipped to the database to be executed there.

To utilize the computing power of modern server platforms, user-defined logic must exploit parallelism provided by the abundance of compute cores. Also, data-specific code and runtime optimizations must be applied to execute application code efficiently. Since SAP's in-memory database technology is evolving from a classical database system to a multi-purpose data analysis engine, runtime optimizations and transparent parallelization must also be applied for non-SQL (procedural) parts of the application code. Focusing on well understood SQL query optimization is not sufficient anymore. Automatic optimization and parallelization of arbitrary procedural parts of application code is required. Algorithms implemented in domain-specific languages or in SQL Script must scale over multiple cores.

One solution is to provide language features that allow application programmers to describe how the in-memory execution engine can optimize and parallelize user code: To address different types of application developers, the system can allow for coding at different levels of abstraction. For example, some application logic can be implemented using graphical tools such as SAP's HANA modeler. Incorporating parallelism hints on this level can be done by providing split/merge operations that split/merge data streams for parallel processing. Again, other parts of the application can be directly implemented in SQL Script, where parallelism can be formulated, for example, using parallel for operations or functional patterns like map/reduce. At the level below, we can provide an infrastructure that generalizes and modularizes existing database-internal algorithms and data structures for easy re-use and re-combination in application code running in the database system.

For convenient development of arbitrary data-intensive logic in SQL Script, such a language will need to evolve from a pure procedural SQL extension to a more complete programming model, supporting features such as modularization, standard libraries, debugging tools, exception handling, etc. The way these challenges are addressed and solved are fundamental, since they will define the way we program future enterprise applications for SAPs in-memory engine.

3 Data Mining Scenario

Let us illustrate the above-mentioned concept of pushing data-intensive operations down into the database with a real-world application from a medical setting. The human proteome represents the collective set of proteins that are expressed by the respective individual's genome at a given time. The proteome is constantly changing, indicating metabolic or environmental conditions like stress or digestion. However, certain proteins (or the lack of them) may indicate a pre-stage to certain diseases, like cancer. The rationale therefore is to analyze the proteomes of two sets of patients — healthy ones and those having a certain disease — and to infer patterns (so-called fingerprints) that distinguish the two groups (but are alike between

patients within the same group). These patterns then can act as biomarkers for the respective diseases.

A prominent way of displaying the proteome is to use mass spectrography. The research institute we are cooperating with uses the MALDI-TOF[1] mass spectrography technique in particular where a blood sample is embedded into a substrate (the so-called matrix) which is then exposed to a laser beam desorbing and charging (ionizing) the contained molecules. These ionized molecules (or fragments of them since they can break apart during the ionization) are then introduced into an electrical field where the fragments are deflected based on their actual mass. A detector measures the number of impacts of fragments w.r.t. their deflection distance thus creating a mass-to-charge histogram. The process is depicted in Fig. 1 and a small portion of a histogram is shown in Fig. 2. The actual ionization and desorption process is carried out many times to actually capture the entire substrate surface leading to a multitude of histograms per blood sample.

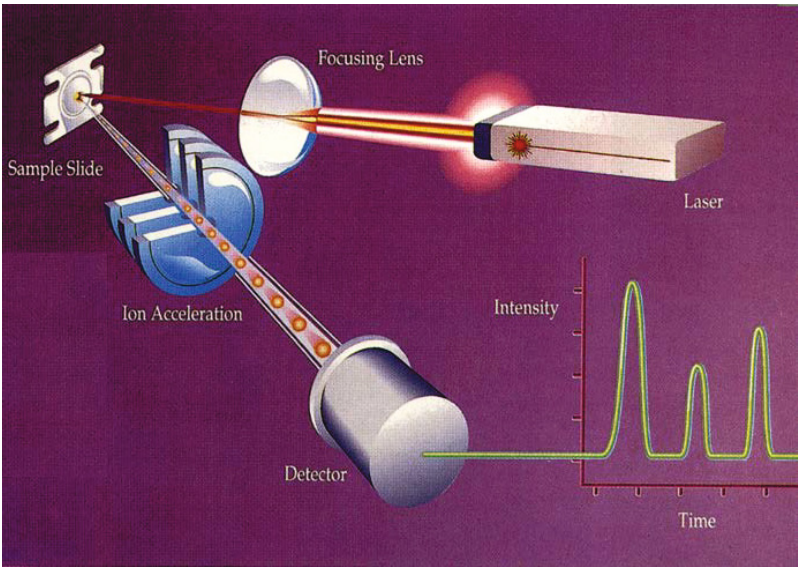


Fig. 1 Elements of a modern MALDI-TOF mass spectrometer: (1) Laser, (2) Sample slide, (3) Acceleration chamber, (4) Drift region, (5) Detector. (picture taken from [1], therein modified and cited from [4]).

3.1 Raw Data

A typical blood sample results in 1,500 to 2,000 histograms with approximately 100,000 histogram bins each. That is, a sample consists of 150 million up to 200 million mass-count pairs (each bin represents a mass and tells the number of fragments

[1] Matrix-assisted Laser Desorption/Ionization, Time-of-Flight mass spectrography.

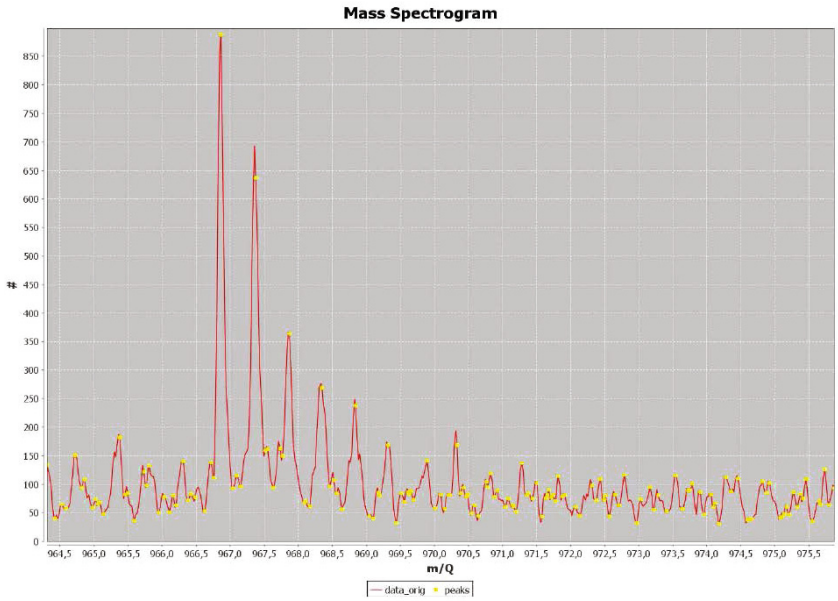


Fig. 2 Portion of a raw spectrum showing pronounced peaks in between noise (maybe containing latent peaks as well). The yellow dots represent seeded local optima that are used for peak picking.

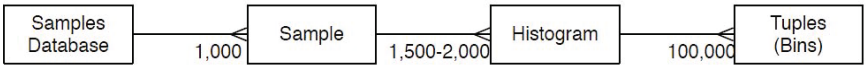


Fig. 3 Quantities of the customer's actual sample database

of that mass that hit the detector). Given as a CSV[2] file, the raw data amounts up to 2 gigabytes of data per sample. In-memory compression reduces this size to just 1 gigabyte inside the HANA database (specific numbers are given in Section 3.4). Figure 3 depicts the (average) cardinalities involved in the data under analysis.

3.2 Analysis Pipeline

The entire analysis pipeline beginning with raw histograms and finally resulting in predictive models consists of four major steps of which we illustrate the peak seeding step in greater detail.

[2] Comma-separated values.

1. **Preprocessing**
   Removal of noise and other systematic errors.
2. **Peak Seeding**
   Identify potential peaks inside the histograms. This is the step for which we illustrate the application of the HANA database.
3. **Peak Picking**
   Taking the seeded peaks and decide which groups form an actual relevant peak to be used for further pattern induction.
4. **Analyses**
   Use the groups of picked peaks to determine whether they can separate the group of healthy patients from the ones with a disease.

The above enumeration also illustrates decreasing amount of data: The raw unprocessed data in step 1 does not contain any model aspect yet and makes up for the largest data volume. Peak seeding returns only those parts of the histogram that might be candidates for actual peaks, thus introducing some degree of semantic meaning while greatly reducing the data (down to 10%–20% of the original size). The remaining two steps further reduce the data size needed to store the actual model. We use this hierarchy to decide which step is most promising to be pushed down into the HANA database to expect the largest gain in performance. Empirical evidence shows that during peak seeding, the largest reduction in data can be achieved, such that we chose this stage for illustrating the application of HANA'a calculation engine.

3.3 Data Flow Model

As mentioned above, the calculation engine of the HANA database acts as the glue that communicates with the different data stores and language layers. The calculation models that are run by the calculation engine can be specified in two ways: Implicitly by issuing a query which is internally translated into a corresponding calculation model, or explicitly by specifying a calculation model as an XML fragment directly to the database. A calculation model is a directed acyclic graph: Root nodes represent data sources while inner nodes denote the procedures that process the data. The edges connect the outputs of a node to the input of the next node (if any). A calculation model is used by the calculation engine to optimize and run the entire set of calculation nodes. Since the XML version of a calculation model is quite verbose, we use the graph in Fig. 4 to illustrate the way the peak seeding is carried out. The sample is imported once into the database and exists as an in-memory column table. Even though the actual data of the histograms are double- and integer-valued, we can achieve a compression ratio of 1:2 compared to the raw CSV data.

For each available CPU, a dynamic view selects a certain histogram from the in-memory column table. Each of these views is fed into a peak seeding node which adds a virtual column denoting which bin of the histogram represents a (potential) peak. The input views contain two columns: a double-valued column for the mass and an integer-valued column for the frequency, that is, the number of fragments

counted with that particular mass value. The left table of Fig. 5 shows an example of such a histogram. Drawn as a continuous function, such a subset of data looks like the plot in Fig. 5. Peak seeding now shall tell for each location whether it is a local maximum or not. For that, the peak seeding procedure adds a new column to the input data where each local maximum is marked with a 1. All other locations are marked with 0 (or ? if it cannot be decided like at the borders of the histogram).

The actual peak seeding is carried out by using a sliding window approach to estimate the first-order derivative. The roots of this sequence of slopes are marked with 1 (and 0 else). The right table of Fig. 5 depicts a potential output. The implementation of this approach is done with an internal language. Upon insertion of the peak seeding procedure code, native machine code is compiled that allows for an efficient execution during runtime.

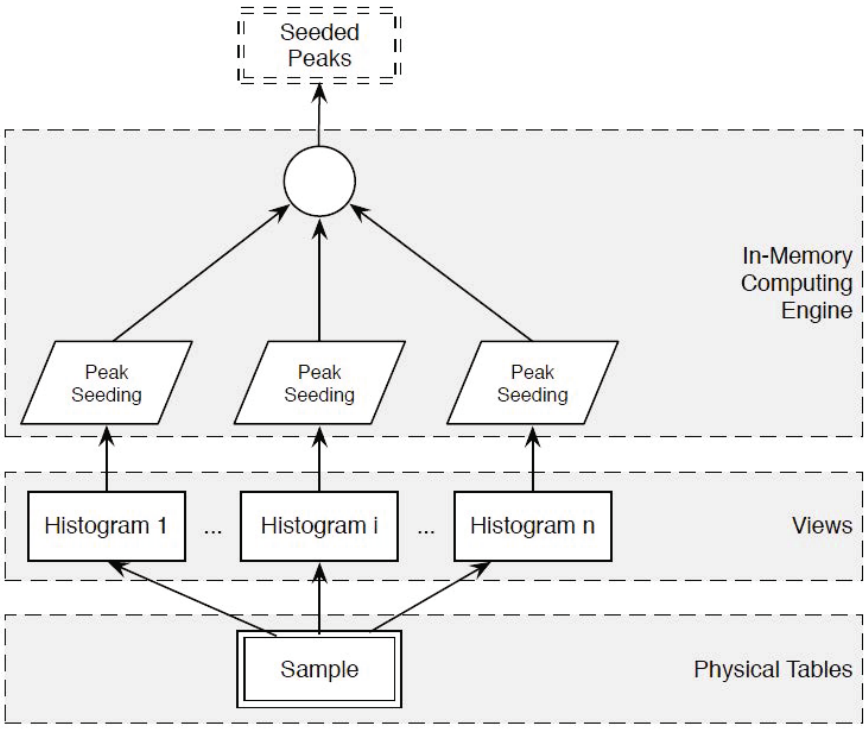


Fig. 4 Logical calculation model for peak seeding. Parallelization is carried out on histogram level. The result (Seeded Peaks) can be either a temporary or materialized table.

3.4 Performance

We carried out measurements with one sample containing 1,680 histograms with an average of 94,086 bins per histogram (minimal bin count: 20,366, max- imal bin count: 105,568). The sample in total contained 158,065,304 mass-freq pairs and

Mass	Freq
23.137728	1
23.140790	3
23.156109	3
23.159172	4
23.162237	2
23.262238	2
23.352617	6
23.355694	8
23.358770	4
...	...

Mass	Freq	IsPeak
23.137728	1	?
23.140790	3	0
23.156109	3	0
23.159172	4	1
23.162237	2	0
23.262238	2	0
23.352617	6	0
23.355694	8	1
23.358770	4	?
...	...	...

Fig. 5 The left table shows an excerpt of a histogram that is fed into the peak seeding procedure as a two-column view. The output (right) contains an additional column denoting whether the respective (mass, freq)-bin is a potential peak (i.e., a local maximum).

had a CSV file size of 2.4 gigabytes. After loading, the in-memory column table consumes only 1.05 gigabytes. Experiments were run on a HP Z600 workstation (2x Intel Xeon X5650 six cores @ 2.67GHz, 24 gigabytes RAM, Suse SLES 11).

We randomly selected 20 consecutive histogram views and ran an equivalent of the calculation plan in Fig. 4 (that is, with 10-fold parallelized peak seeding). The number 20 was empirically determined to on the one hand equally load all cores and on the other hand deliver running times that were long enough to get a stable average. The execution times for the entire model (selecting input views, running the peak seeding and returning the result views) averaged between 60 ms–70 ms for 20 histograms. We conducted no representative tests as to how the application behaves under heavy concurrent requests as the sketched proteomics scenario is embedded in an analysis pipeline where a large number of users will be relevant only later in the process, namely when consuming the predictive content based on the results from the algorithms discussed above.

4 Summary and Next Steps

We presented architectural and technological insights into SAP's HANA database platform and derived research challenges for future enterprise application development. Additionally, a real-world application was introduced where we applied new programming paradigms. Early results show great potential for changing the way data analysis application will be designed and consumed.

In the example above, we chose the peak seeding stage for illustration as it was the most data-intensive step. However, considering the large number of samples in a typical proteome library, the latter steps are currently also being pushed into the database kernel. That is, model induction and prediction are planned to be carried out entirely inside the database, too.

References

[1] Conrad, T.O.: New statistical algorithms for the analysis of mass spectrometry time-of-flight mass data with applications in clinical diagnostics. PhD thesis, Freie Universität Berlin (2008)
[2] Färber, F., Cha, S.K., Primsch, J., Bornhövd, C., Sigg, S., Lehner, W.: SAP HANA database: data management for modern business applications. SIGMOD Rec. 40(4), 45–51 (2011)
[3] Jaecksch, B., Faerber, F., Rosenthal, F., Lehner, W.: Hybrid Data-Flow Graphs for Procedural Domain-Specific Query Languages. In: Bayard Cushing, J., French, J., Bowers, S. (eds.) SSDBM 2011. LNCS, vol. 6809, pp. 577–578. Springer, Heidelberg (2011)
[4] MALDI-TOF Mass Analysis (2011), http://www.protein.iastate.edu/maldi.html (last accessed on June 19, 2012)
[5] Plattner, H.: A common database approach for OLTP and OLAP using an in-memory column database. In: Proc. 35th SIGMOD Intl. Conf. on Management of Data, SIGMOD 2009, pp. 1–2 (2009)
[6] Plattner, H., Zeier, A.: In-Memory Data Management: An Inflection Point for Enterprise Applications. Springer, Heidelberg (2011)

Computational Intelligence in Air Traffic Management

Annette Temme, Ingrid Gerdes, and Roland Winkler

Abstract. The demand for increasing airport capacity combined with many constraints as well as the complexity of the data itself leads to the use of heuristic methods from the computational intelligence domain. More specifically, the focus in this paper is on how (fuzzy) clustering methods and evolutionary algorithms are applied on various aspects of the Air Traffic Management domain. Fuzzy clustering techniques have been used for data evaluation and pre-processing. One task is the identification and correction of noise and outliers in radar tracks as a pre-processing step. In addition, clustering has been applied to identify general flight routes in retrospective analysis tasks as well as to generate fuzzy rules, thus verifying or complementing expert knowledge regarding transfer passenger movements. Evolutionary algorithms are used to assist air- and ground traffic controllers. Namely in Rogena (free ROuting with GENetic Algorithms) for route planning and TRACC (Taxi Routes for Aircraft: Creation and Controlling) for ground movement planning. Both systems create conflict free routes for aircraft which are suggested to the air- and ground traffic controllers, respectively.

1 Introduction

The increasing human mobility in the last decades led to growing traffic rates at most major airports around the world. These airports are usually located in urban areas where airport expansions are hardly suitable. Finding alternative solutions for increasing the airport capacity is a very complex task and must be done for each

Annette Temme · Ingrid Gerdes
Institute of Flight Guidance, German Aerospace Center, 38108 Braunschweig, Germany
e-mail: {annette.temme,ingrid.gerdes}@dlr.de

Roland Winkler
Leibniz Institute for Astrophysics, 14482 Potsdam, Germany
e-mail: roland.winkler@gmail.com

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 285–297.
springerlink.com

airport individually. The ultimate goal is to improve the airport's performance. To achieve that, the airport is on the one hand analysed in its current situation and limiting elements are identified in retrospective evaluations. Relevant data for such analysis can be flight information data and / or radar data combined with additional information such as time, aircraft size and type, weather, airline dependent information, airport layout, and airport regulations. On the other hand, optimisation techniques are applied to controller assistance systems. The tasks in the Air Traffic Management (ATM) domain are usually complex and include additional data related challenges like a very high number of attributes, limited sensor accuracy, missing values in data sets, and in some cases even manual data gathering. The demand for increasing airport capacity combined with many constraints as well as the complexity of the data itself leads to the use of heuristic methods from the computational intelligence domain. More specifically, the focus in this paper is on how (fuzzy) clustering methods and evolutionary algorithms are applied on various aspects of the Air Traffic Management domain. Besides the applications described exemplary here, several approaches to apply fuzzy clustering and soft computing techniques to traffic problems and aerospace applications in general have been described in the literature, see e.g. [3, 4, 6, 1, 14]. Fuzzy clustering techniques have been used for data evaluation and pre-processing by the authors. One task has been the identification and correction of noise and outliers in radar tracks as a pre-processing step [12]. Fuzzy clustering has also been applied to generate fuzzy rules, thus verifying or complementing expert knowledge regarding transfer passenger movements [16]. In addition, clustering has been applied to identify general flight routes [13] and correct radar tracks [19] as well as part of airport performance evaluation [18]. The last two are exemplarily presented in this work. In addition, two systems based on evolutionary algorithms are shown: Rogena (free ROuting with GENetic Algorithms) for route planning and TRACC (Taxi Routes for Aircraft: Creation and Controlling) for ground movement planning. Both systems create conflict free routes for aircraft which are suggested to the air- and ground traffic controllers, respectively. The first two applications presented here are designed for retrospective analysis. The systems Rogena and TRACC are implemented in test environments for air traffic controllers. Before such real-time systems are introduced at existing airports, comprehensive safety tests and evaluations have to be performed. Beyond the scope of this paper but even more complicated are technological modifications of aircraft, because the certification procedures are very restrictive here.

2 Fuzzy Clustering

Fuzzy clustering techniques are designed to find a suitable fuzzy partition for a given data set. For a fuzzy partition a data object is not necessarily assigned to a unique class or cluster, but has membership degrees between zero and one to each cluster. The use of fuzzy clustering algorithms has several advantages w.r.t. classical (crisp) clustering:

- The membership degrees give information about the ambiguity of the classification.
- Fuzzy clustering can adapt to noisy data and classes that are not well separated.
- Since most fuzzy clustering approaches are based on optimising an objective function, membership degrees represent continuous parameters so that a continuous optimisation problem has to be solved.
- Fuzzy clustering can be applied to learning fuzzy rules from data.

The set of cluster parameters, that determine the size and the shape of a cluster, depends on the specific application field. In the ATM domain, fuzzy clustering is mainly used as an explorative data analysis method, especially for unsupervised classification tasks. For an overview on fuzzy clustering see for example [11].

2.1 Fuzzy Clustering in Airport Performance Estimation

One aim in optimising air traffic management is to use the given resources more efficiently and thereby increase the performance of airports over time. That implies that there must be some sort of performance measure for airports in order to evaluate air traffic management systems. The methodologies applied in such a system have to be independent from an actual airport's layout in order to generate comparable results for all larger European airports. One part of such a measuring system has been developed by DLR in connection with the Performance Review Unit (PRU) of EUROCONTROL [5].

In particular, it is measured how well the airspace around an airport for arriving aircraft is controlled. The two most important indicators for the performance are the mean travel time and the variance of the travel time for the last 100 Nautical Miles (NM) before landing. The mean travel time has to be as low as possible to reduce the mean flight time and the variance should be as low as possible to make flight times more predictable. More predictable flight times would mean, that flight plans can be condensed which directly increases the capacity of the traffic system.

The flight time for the last 100 NM depends strongly on the direction of approach (i.e. the origin of the aircraft) and the current runway configuration because it determines the aircraft's landing direction. For example an aircraft that can land in the same direction as it is approaching the airport has to fly a much shorter distance than another one that has to surround the airport before landing. The runway configuration is almost entirely determined by the direction and speed of the wind. Obviously, both factors (direction of wind and origin of an aircraft) are out of control of the airport and must be taken into account for calculating the performance indicators.

The performance indicators are calculated using radar tracks of aircraft which indicate the location of an aircraft once per minute. The data contains all civil aircraft movements in the European airspace, covering 7 days in May 2005. For each aircraft, the entry point into the 100 NM radius around an airport is used to determine its local origin. In Fig. 1, two example data sets are presented, which show

the first entry observation. The airport is located in the centre and the black dots represent the aircraft approaching the airport. Some of the groups are too small or some aircraft approach the airport from an unusual angle, these have to be regarded as noise because they do not provide sufficient data to make statistically reliable measurements.

The number of clusters is unique for each airport and the number should be determined automatically. The standard FCM algorithm [2] does not provide the capability of calculating these properties. Therefore, we use a special FCM like algorithm [18] to cluster these data sets, which has an additional term of repulsion between the prototypes in order to prevent them being too close together. To determine the number of clusters, an overestimated number of prototypes is used at the beginning and those prototypes that did not represent a sufficient amount of data objects are removed after the clustering has finished. In a second step, the obtained and yet (due to the repulsion) distorted prototype positions are used to initialise a standard FCM algorithm. The result of this two-step clustering process is presented in Fig. 2. The prototypes are represented as circles and the grey shading indicates the cluster assignment (which is crisp here to make it easier to recognize visually). Black data objects far away from any prototype represent the noise cluster. Both examples were calculated using the same parameters which shows the usefulness of the approach.

The cluster information is then used to calculate for each runway configuration (which is determined in a similar manner) and each approach cluster the mean travel time and variance in travel time. The resulting values are used in combination with several other key performance indicators, to determine the performance of an airport.

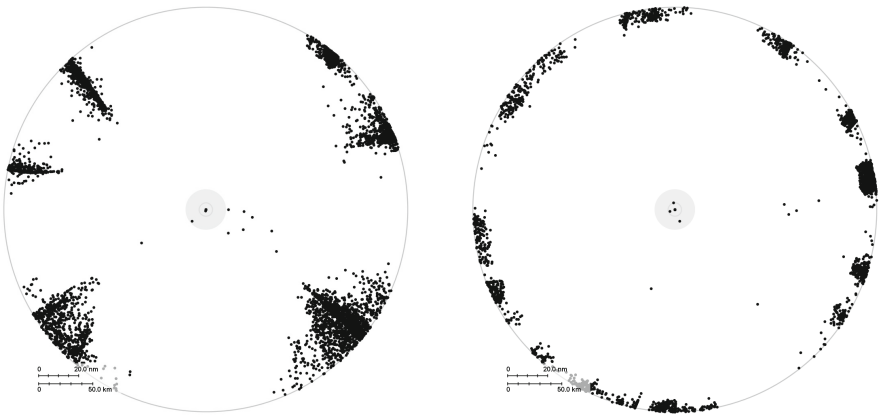


Fig. 1 Two typical data sets of different airports. Each dot represents the first measured position of an aircraft inside the 100NM radius around an airport.

2.2 DB Scan in Aircraft Position Estimation

One general problem in avionics is to determine the location of an aircraft. The aim of the approach presented here is to include the developed method into a system for retrospective movement analysis at an airport. Therefore, the technology to determine the location has to be as independent of the aircraft as possible. GPS is not such a system for two main reasons. First, it cannot be guaranteed that all aircraft have a GPS transponder and can broadcast their location. Second, GPS could fail, either for technical problems or political intervention. Here, only systems that are independent of the technology of the aircraft can be used.

In the air, en-route radar systems have relatively slow moving antennas. This leads to a low accuracy in determining an aircraft's position and in turn enforces larger separations. However, on the ground, the location of aircraft must be much better determined because the aircraft are much closer together than in the air. The surface radar is not capable of providing the desired level of accuracy. In addition, it cannot always be guaranteed to have a free line of sight to each aircraft.

A complementary system MLAT (Multilateration) detects the origin of the transponder signal of an aircraft (Aircraft without a transponder are not allowed to enter commercial used airspace) by measuring the signal time delay w.r.t. several receiving antennas on the airport area. In combination with the surface radar, the location of an aircraft can be detected in the order of a few 10 meters. To further improve the quality, the current location and measured speed of the aircraft is used to estimate a future location which in combination with the new measurement improves the accuracy to a few meters.

If the speed of the aircraft becomes very slow, the prediction of the next position due to the current speed of the aircraft becomes meaningless. The algorithm does not detect this problem and reports a widely oscillating location of an aircraft around its true location. In Fig. 3, two examples of such a problematic measurement are

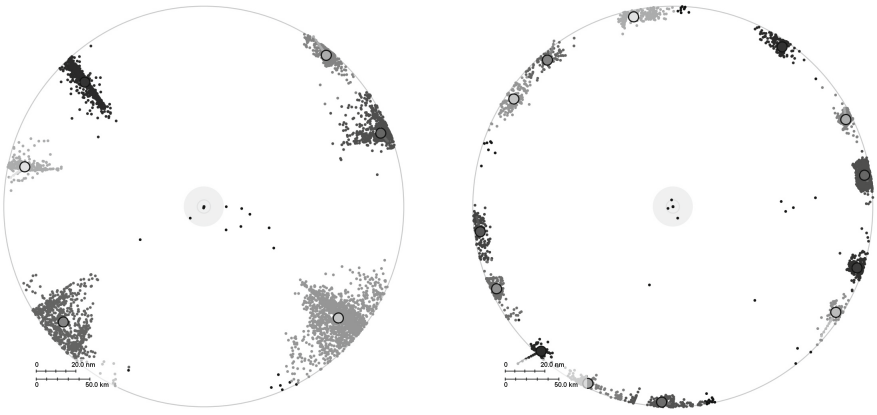


Fig. 2 The clustering result of the two above presented data sets. Black dots indicate the noise cluster.

presented. It is necessary to remove these errors in order to detect when and where an aircraft stops. The exact stop times are important factors for future planning and also for automatic evaluation of the current situation at the airport. Fraport and DLR have developed a system to remove the local inaccuracy due to the prediction algorithm for low aircraft speeds. This will be applied to the tool S.O.D.A. [17] developed by Fraport. Movement models like addressed in [10] cannot be well applied on the original data due to the inaccuracy of the measurements. For further analysis however, an adapted version for aircraft moving on the ground might be well suited.

Each observed track is analysed separately for local 'clouds' of position observations. If such a cloud occurs, it is likely to be a bad observation because an aircraft does not drive around widely near one spot. The DBScan algorithm [7] is used to separate 'good' from 'bad' observations and to estimate their respective correct

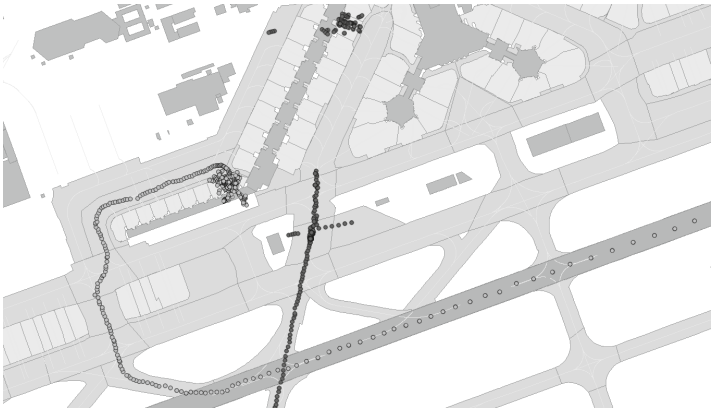


Fig. 3 Two problematic observations of aircraft location on the Frankfurt airport. The background shows parts of the ground structure of the airport.

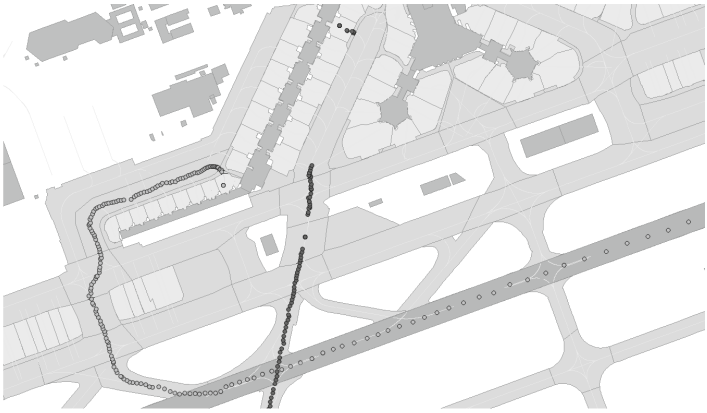


Fig. 4 The two problematic tracks from above, this time with corrected 'bad' observations

location. The process to handle all exceptional cases is rather complicated, but the basic idea follows the following three step approach:

- In the first step, DBScan is used on the 3-dimensional location and time values of each observation, with a low number of core points and a small core radius. This divides each track into a subset of shorter tracks, separated gaps in space or time. Many observations in one group means there are no gaps and therefore, it is likely to be a group of good observations. Such groups are excluded from later corrections.
- In a second step, clouds are found by applying DBScan again, on the 2 dimensional spatial locations, with a larger number of core points and slightly larger radius. This ignores long strings of observations and detects dense clouds. For each cloud separately, the centre of mass is used as the estimated true position of the aircraft.
- In the third step, all observations belonging to small groups (or the noise cluster) are corrected to the centre of mass of the clouds that are detected in the second step. Each cloud defines a time frame for which the aircraft has stopped and this time frame is used to determine which 'bad' observation belongs to which cloud.

The result of this correction process is shown in Fig. 4. With these corrected observations, a lot of higher level analysis can be done.

3 Evolutionary Algorithms

The theory of evolutionary algorithms was developed as artificial evolution of a population of science problems in form of chromosomes [9]. These chromosomes consist of genes which code different aspects of a possible problem solution. They are mainly used for optimisation problems with a large solution space. The approach is adopted from nature with exchange (crossover) or mutation of genetic material with the goal of creating better solutions over time. The selection pressure is created by an evaluation formula depending on important optimisation parameters. It influences the chance of survival by a selection probability for each chromosome which is directly connected to the related evaluation value.

A standard evolutionary algorithm starts with the random creation of a population of binary coded chromosomes as possible solutions which is then evaluated with the previously defined evaluation formula and the chromosomes for the next generation are selected randomly. After applying the crossover and mutation operators with a small probability to each chromosome of the population, the sequence of evaluation, selection, and application of operators starts again until a stopping criterion is fulfilled. Especially in the beginning particularly fit chromosomes can dominate the population in the selection phase (so-called *super individuals*). This can lead to a lack of diversification in the population and in the end to a loss of good solutions. For coping with these super individuals a modified genetic algorithm called modGA was developed by Michalewicz [15]. Instead of selecting all chromosomes and applying the genetic operators to them three different groups of chromosomes

are created. The chromosomes of the first group have to be different and remain unchanged, the second group undergoes the crossover and the third group the mutation operator. With introducing the first group a minimum of diversity is maintained.

3.1 ROGENA (Free ROuting with GENetic Algorithms)

Using predefined flight routes does not scale with increasing traffic. All aircraft have to maintain a prescribed horizontal minimum separation to other aircraft which depends on the position of the aircraft (glide path of the airport or remaining airspace). In the first case, aircraft have to use the so called wake vortex separation depending on the weight of the engaged aircraft and in the second case a horizontal separation in the range of 5 to 10 NM and 1000 to 2000 feet vertical depending on the flight level has to be maintained. Furthermore, there are restricted areas which cannot be crossed.

Today all aircraft have to use prescribed standard flight routes, which often force the aircraft to fly a detour to reach its destination. This increases not only the flight time and the cost (fuel, staff) unnecessarily but leaves the main part of the airspace unused. For manual control of the airspace the advantage of standard routes for controllers is the limited number of crossing points between routes.

Nevertheless, more flexibility in creation of routes is needed but in this case the controller will need technical support because of the resulting much more complicated airspace structure. With ROGENA such a technical support was implemented and tested. The purpose of this tool is the creation of short and conflict free approach routes with applicable descent and speed profiles inside the TMA. Because it is common practice to use 'First Come, First Serve' in the assignment of flight routes and keeping the number of route changes caused by non-compliances of other aircraft as low as possible, not all aircraft routes are optimised at once but step by step. Only those are reassigned, which have violated their assigned route during the flight-phase.

Before it is possible to apply an optimisation technique to a problem it is necessary to formalize this problem. In case of ROGENA a square of the airspace around an airport of 200 by 200 NM was selected. Routes are described as a sequence of waypoints (links between waypoints) consisting of x-, y- and z-coordinates as double values together with data for speed and the used flight level. The number of used waypoints can differ, but start and end points are predefined. Because no curves are used for the routes it is necessary to assure a minimal angle between consecutive links of 90 degrees. Within ROGENA the links of the routes are handled as vectors in space. Therefore, the position of an aircraft is defined by the following formula:

$$\mathbf{p} = \mathbf{w_i} + \lambda (v_i + \Delta v_i t) t (\mathbf{w_{i+1}} - \mathbf{w_i})$$

where w_i = waypoint i, v_i = speed at waypoint i, Δv_i = speed change per time unit $t\,[m/s^2]$. λ in $[1/m]$ is the reciprocal of the length of the directional vector. For a possible application of the routes created by ROGENA it is essential to comply to feasible descent and climb profiles which depend on aircraft type, flight level and

stall speed (speed of airflow breakaway). Standard speed profiles can be calculated from aircraft procedure models of the EUROCONTROL BADA-database [8]. Speed differences between successive way points are processed linear. Restricted areas are implemented as polygons with a lower and a top level as well as a life time.

As input data for traffic a flight plan with flight id, type of aircraft, enter time for the observed airspace, landing time and weight class (wake vortex class) for each aircraft is needed.

Another formalisation is necessary for the holding patterns for aircraft which are used for increasing punctuality or avoiding conflicts with other aircraft. This holding pattern is an oval structure with a prescribed size depending on aircraft speed and flight level. They are approximated by a hexagon with two long and four small legs.

There are two different types of possible conflicts which have to be observed by ROGENA. The first type are conflicts between moving aircraft, the second type are conflicts between aircraft and restricted areas. The conflict calculation between aircraft is carried out by applying the rules of vector analysis. It is done by testing for easy cases first (e.g. both flights are parallel) and otherwise calculating the point of time where the minimum distance for each pair of links from two different aircraft is reached. These points of time are calculated as extreme values of the distance equation

$$dis^2(t) = (\sum_{i\in\{1\ldots3\}} (a_i - c_i + (v_1 t + \Delta v_1 t^2) b_i - (v_2 t + \Delta v_2 t^2) d_i)^2 = 0$$

for aircraft 1: $(\mathbf{a} + (v_1 + \Delta v_1 t)t\mathbf{b}$, aircraft 2: $(\mathbf{c} + (v_2 + \Delta v_2 t)t\mathbf{d}$. Solving this equation leads to a maximum number of three extreme values. If these extreme values are reached within the time the aircraft occupies the observed link, this value is used for calculating the closest reachable distance, otherwise the closest time value on the link (boundary point). For a correct separation between two aircraft not only the lateral separation should be guaranteed but the vertical separation also. In case of conflicts the time of the first height violation is calculated for a conflict check.

For the conflict detection between aircraft and restricted areas the intersection points between all aircraft and area links for the two-dimensional area are calculated and a comparison between the intersection time with time of validity for the restricted area is carried out. In case of intersections it is tested whether the critical part of the aircraft route hits the flight levels occupied by the restricted area.

The evolutionary algorithm of ROGENA is based on the ideas of modGA. However, problem specific modifications of the modGA have to be included for a significant improvement of performance (e.g. storing of best five routes of every generation automatically). The size of the population is set to 60 chromosomes (or routes) and 20 chromosomes for each of the three groups. Instead of using binary coded chromosomes double values where used for the waypoint information.

Start-population: The chromosomes of the start population are created with predefined start- and endpoints and speeds and a random number of waypoints between them. The location of the next random waypoint is determined by evenly dividing the connection from the last selected point (e.g. start point) to the endpoint by the

number of remaining waypoints. Then a deviation from x- and y-coordinates using a normal distribution is carried out. With this approach the new route runs close to the direct connection.

Evaluation formula: The evaluation function for the chromosomes consists of values for the length, lack of separation, penalty points for intersecting restricted areas, penalty points for using incorrect climb and descent profiles, and penalty points for using to narrow flight angles between successive waypoints. These five parts are combined with generation dependent weight factors which reflect the changing importance of the different parts of the evaluation function with increasing generation number.

Selection of chromosomes: For the selection of chromosomes for the first group it is necessary to define the term *different* because the use of double values for the coordinates can lead to many nearly similar routes similar to *super individuals*. For ROGENA *different* means either a different number of waypoints or a certain distance between way points at the same position of the chromosome. Furthermore, a generation dependent maximal allowed evaluation value is calculated excluding worse chromosomes from taking part in the selection. The selection probability for route i is then directly proportional to the evaluation value: $prob(i) = \left(\frac{maxEval - eval(i)}{sum(maxEval eval(j)}\right)$ with $prob(i) = 0$ for routes with evaluation value greater than maxEval.

Crossing of chromosomes: Because the chromosomes can differ in the number of waypoints it is necessary to determine the minimum number of waypoints *minNr* of two routes. Then two indices as crossover points are selected by random and the nodes between these indices are exchanged and the standard speeds (see start population) are reassigned. The flight levels for the exchanged nodes are linearly determined to the difference between the last waypoint before and the first behind the crossed part of the route.

Mutation of chromosomes: For ROGENA three different types of mutation are applied with a small probability:

- Mutation of waypoints (normal distribution for the coordinates of the old waypoint),
- Mutation of the number of waypoints.
- Mutation of a flight level.
- Introducing a holding pattern.

Tests with different scenarios have shown that ROGENA is able to create conflict free routes which are also short and close to the direct connection and conform to the conditions for speeds and climb and descent rates without using prescribed standard routes. Therefore, it is possible to include much more of the today unused airspace for a relief of the more and more overloaded traffic situation around airports.

3.2 *TRACC (Taxi Routes for Aircraft: Creation and Controlling)*

TRACC is designed for the use within the DLR Tower Simulator (real-time). The focus of TRACC is the creation of time-based and conflict free taxi-routes on the basis of a group of standard routes, which can be changed if required. Furthermore, time constraints (e.g. take off time) are taken into account. A screen-shot of TRACC is shown in Fig. 5.

For each new flight the normal standard route used at the observed airport is applied as a starting point together with a standard speed profile. In the next step this route is handed over to an evolutionary algorithm which tries to optimise the speed profile with respect to the time constraints and the movements of other aircraft (as far as they are already scheduled). In case of conflicts which cannot be solved by using the standard route, a second evolutionary algorithm is started which is able to create completely new routes without respect to standard operations. Both evolutionary algorithms are based on the same algorithm type and conflict detection algorithms (2-dimensional for TRACC) as ROGENA but with several necessary additions. Because possible taxi routes have to adhere to the taxiway system the possibilities for the creation of routes are limited. Furthermore, the genes of the chromosomes consist of nodes of the underlying node-link system (including circular arcs) instead of coordinates. Exchanging parts of chromosomes does now result in the necessity to repair the chromosomes by adding the connections between the old and the new parts of the route. The same is true for the mutation operator and has led to a substitution of a part of a chromosome by a randomly selected new node

Fig. 5 Airport with underlying node-link structure

and the necessary new connections. A function for the removal of circular driving parts is added together with control functions for taxi speeds.

4 Conclusion

In this contribution, we have shown some examples where methods of the computational intelligence domain have been successfully applied to problems from the air traffic management domain. Depending on the type of the problem, e.g. data analysis or optimisation tasks, a suitable method has to be chosen. For the analysis of imprecise radar data, especially fuzzy clustering techniques have proven their usefulness. Evolutionary algorithms allowed to demonstrate that the use of more flexible routes will be possible with suitable assistance systems. In addition, with evolutionary algorithms, it was possible to develop a taxi route generator that is able to cope with numerous boundary conditions and thus calculates routes comparable to those that would be chosen by a traffic controller.

References

[1] Beasley, J., Sonander, J., Havelock, P.: Scheduling aircraft landings at London Heathrow using a population heuristic. Journal of the Operational Research Society 52, 483–493 (2001)
[2] Bezdek, J.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)
[3] Blumel, A.L., Hughes, E.J., White, B.A.: Design of robust fuzzy controllers for aerospace applications. In: Davé, R.N., Sudkamp, T. (eds.) 18'th Int. Conf. of the North American Fuzzy Information Processing Society - Nafips, New York, USA, pp. 438–442 (1999)
[4] Cafiso, S., Cutello, V.: A fuzzy model for road accidents analysis. In: Davé, R.N., Sudkamp, T. (eds.) 18'th Int. Conf. of the North American Fuzzy Information Processing Society - Nafips, New York, USA, pp. 139–143 (1999)
[5] Performance Review Commission, ATM Airport Performance (ATMAP) Framework: Measuring Airport Airside and Nearby Airspace Performance. Eurocontrol (2009)
[6] Das, S., Bowles, B.A.: Simulations of highway chaos using fuzzy logic. In: Davé, R.N., Sudkamp, T. (eds.) 18'th Int. Conf. of the North American Fuzzy Information Processing Society, Nafips, New York, USA (1999)
[7] Ester, M., Kriegel, H.P., Jörg, S., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: 2nd International Conference on Knowledge Discovery and Data Mining, pp. 226–231. AAAI Press (1996)
[8] Eurocontrol, User Manual for the Base of Aircraft Data (BADA) Release 2.3. Eurocontrol (1995)
[9] Gerdes, I., Klawonn, F., Kruse, R.: Evolutionäre Algorithmen. Vieweg, Wiesbaden (2004) (in German)
[10] Guerrero, J., García, J., Molina, J.: Air traffic control: a local approach to the trajectory segmentation issue. Trends in Applied Intelligent Systems, 498–507 (2010)
[11] Höppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy Cluster Analysis. Wiley, Chichester (1999)

[12] Keller, A.: Fuzzy clustering with outliers. In: Whalen, T. (ed.) PeachFuzz 2000, 19th International Conference of the North American Fuzzy Information Processing Society, Nafips, Atlanta, pp. 143–147 (2000)
[13] Keller, A.: Objective function based fuzzy clustering in air traffic management. PhD thesis, Otto-von-Guericke-Universität Magdeburg, Magdeburg, Fakultät für Informatik (2002)
[14] Korn, B.: Fuzzy clustering techniques in autonomous sensor-based landing systems. In: FSCS 2006 - Symposium on Fuzzy Systems in Computer Science (2006)
[15] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin (1992)
[16] Temme, A., Kruse, R.: A fuzzy rule system describing transfer passenger movements. In: Di Nola, A. (ed.) Soft Computing: A Fusion of Foundations, Methodologies and Applications, vol. 10, pp. 917–923. Springer, Heidelberg (2006)
[17] Wendeberg, S.: Kapazitätsplanungen eines Verkehrsflughafens. In: Kolloquium Flugführung, Braunschweig (2012) (in German)
[18] Winkler, R., Rehm, F., Kruse, R.: Clustering with repulsive prototypes. In: Advances in Data Analysis, Data Handling and Business Intelligence. Studies in Classification, Data Analysis and Knowledge Organization, vol. 2, pp. 207–215. Springer (2010)
[19] Winkler, R., Temme, A., Bösel, C., Kruse, R.: Clustering radar tracks to evaluate efficiency indicators. In: Proc. of the second ENRI Workshop on ATM and CNS, Tokyo, Japan, pp. 71–94 (2010)

Part VI
Closing Remarks

About Rudolf Kruse and His Research Group on Computational Intelligence

Christian Moewes and Andreas Nürnberger

Abstract. The preceeding chapters contain original contributions on the occasion of Rudolf Kruse's 60th birthday. These papers are categorized in the four research areas to which Rudolf Kruse and his research group contributed to, i.e. fuzzy data analysis, hybrid intelligent systems, uncertainty in knowledge-based systems, and intelligent data analysis. Each topic spans one part of this book whereas the corresponding papers are ordered alphabetically by the last name of the first author. The fifth part comprises papers that describe the application of computational intelligence methods to real-world data analysis problems. This gives some more historical insights into the research works of Kruse and his group.

Rudolf Kruse obtained his diploma (Mathematics) degree in 1979 from University of Braunschweig, Germany, and a PhD in Mathematics in 1980 as well as the venia legendi in Mathematics in 1984 from the same university. Following a short stay at the *Fraunhofer Gesellschaft*, in 1986 he joined the University of Braunschweig as a professor of computer science. Since 1996 he is a full professor at the Faculty of Computer Science of the University of Magdeburg where he is leading the computational intelligence research group.

Rudolf is the mentor of 20 doctorates and habilitants. He supervised more than 300 undergrate and graduate students. Since decades, he is giving lectures about a broad topic of computational intelligence methods. His group published many student textbooks in German and

Christian Moewes · Andreas Nürnberger
Faculty of Computer Science, Otto-von-Guericke University, 39106 Magdeburg, Germany
e-mail: {cmoewes,andreas.nuernberger}@ovgu.de

C. Moewes et al. (Eds.): Computational Intelligence in Intelligent Data Analysis, SCI 445, pp. 301–304.
springerlink.com

English on many aspects of computational intelligence. Some of the most known English textbooks are about fuzzy systems [10] and neuro-fuzzy systems [14]. This eventually led to an English student textbook entitled *Computational Intelligence* [12] which appeared in the Springer series *Textbooks in Computer Science* in 2012.

Rudolf has coauthored more than 350 referred papers and 40 books. According to Google Scholar, he has more than 8,000 citations and currently an h-Index of 41. Kruse is associate editor of 10 scientific journals. He is a fellow of the *International Fuzzy Systems Association (IFSA)*, fellow of the *European Coordinating Committee for Artificial Intelligence (ECCAI)* and fellow of the *Institute of Electrical and Electronics Engineers (IEEE)*.

The first research area where Rudolf Kruse and his group contributed to is *fuzzy data analysis*. Its aim is to analyse both *crisp data using fuzzy methods* and *fuzzy data using standard methods*, e.g. statistics. Kruse and his group published the first monograph about fuzzy statistics [7]. Already in Braunschweig, he organized workshops on that topic, e.g. *Fuzzy Systems '93 – Management of Uncertain Information* [11]. Even though it was the first of his interests, it is still lively discussed today, e.g. in the just finished COST Action IC0702 "SoftStat" which focused on the combination of statistics and soft computing.Also, this year from October 4 to 6, Michael Berthold and Rudolf Kruse chair the 6th International Conference on Soft Methods in Probability and Statistics in Konstanz, Germany where fuzzy data analysis will be a major topic. Regarding the industry, Rudolf inspired the development of many applications especially dealing with fuzzy clustering [5], e.g. at the *German Aerospace Center*.

His second research area focuses on *hybrid intelligent systems*. There, typically computational intelligence methods are intelligently combined, e.g. a fuzzy system with an artifical neural network. Such a neuro-fuzzy system [14] encodes the fuzzy rules into the network and uses neural network learning algorithms. They can be used in control [13], classification and function approximation. For the development of a fuzzy idle speed controller together with Volkswagen AG [6], Kruse and his group received the outstanding paper award in *IEEE Transactions on Fuzzy Systems* in 1996. Working together with VW, one of Kruse's student used such a hybrid intelligent system to design an intelligent gear system in the VW New Beatle [16].

Rudolf Kruse also contributed to the research field *uncertainty in knowledge-based systems*. Many contributions have been made by Rudolf Kruse's group to handle uncertainties, vagueness, incompleteness or partial inconsistency. Kruse's monograph [9] on this topic from 1991 was one of the first monographs on Bayesian networks. Nowadays, his monograph about learning and representing graphical models has been already extended and published in a second edition [1]. The ESPRIT Basic Research Action 3085 (named *Defeasible Reasoning and Uncertainty Management Systems (DRUMS)* focused on these methodologies. Rudolf Kruse's research group was one of 11 European ones that participated in that project. In 1992, he established a forum for these groups with the *European Conferences on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*.

The first conference in Marseille, France attracted 140 participants. The conference proceedings are still published at Springer-Verlag every other year. During the time of the DRUMS project, 4 books about this topic had been coeditored by him [8, 2, 15, 3]. Similarly to the DRUMS project, Rudolf Kruse and Giacomo Della Riccia, the last doctorate of Norbert Wiener who is the originator of cybernetics, biannually invited internationally renowned researchers to the picturesque Palazzo del Torso of the Centre International des Sciences Mécaniques (CISM) in Udine, Italy. The revised versions of workshop papers were always published in the series *CISM Courses and Lectures* which resulted into 7 books featuring varying topics.

His group in cooperation with Dornier implemented the most like first Bayesian network in Germany. The research [4] on this topic (see also page 153) let to a successful outsourced company where 5,000 Bayesian networks are used on a daily basis.

Most recently, Rudolf Kruse is mainly interested in *intelligent data analysis*. Here, his focus is on the development of new learning methods and temporal data analysis. This book offers a great collection of research questions dealing with this topic. Most applications he has based his research on stem from collaborations with rating agencies at the *Deutsche Sparkassen- und Giroverband (DSGV)*, Europe's largest automobile club *Allgemeiner Deutscher Automobil-Club e.V. (ADAC)*, Daimler, British Telecom (BT), Siemens, Commerzbank and medical institutes at the University of Magdeburg. The interactive data mining platform *Information Miner* has been established during these cooperations and is still in the main focus of ongoing software development in his group today. It is not only used in lectures to visualize and enhance intelligent data analysis, it is also presented at the international exhibition *CeBIT* in Hannover, Germany every year since 2005. Its presentation enables a lively technology transfer from Rudolf Kruse's working group to both industries and the public sector. He shows the usefulness of the methods by consulting companies that typically deploy parts of his tools to solve real-world problems.

References

[1] Borgelt, C., Steinbrecher, M., Kruse, R.: Graphical Models: Representations for Learning, Reasoning and Data Mining, 2nd edn. Wiley Series in Computational Statistics. John Wiley & Sons, Inc., Chichester (2009)

[2] Moral, S., Kruse, R., Clarke, E. (eds.): ECSQARU 1993. LNCS, vol. 747. Springer, Heidelberg (1993)

[3] Gabbay, D.M., Kruse, R. (eds.): Handbook of Deafesible Reasoning and Uncertainty Management Systems, Abductive Reasoning and Learning, vol 4. Kluwer (2000)

[4] Gebhardt, J., Klose, A., Detmer, H.: Graphical models for industrial planning on complex domains. In: Riccia, G.D., Dubois, D., Kruse, R., Lenz, H.J. (eds.) Decision Theory and Multi-Agent Planning. CISM Courses and Lectures, pp. 131–143. Springer, Heidelberg (2006)

[5] Höppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy Cluster Analysis: Methods for Classification, Data Analysis, and Image Recognition. John Wiley & Sons, Ltd., Chichester (1999)

[6] Klawonn, F., Gebhardt, J., Kruse, R.: Fuzzy control on the basis of equality relations with example from idle speed control. IEEE Transactions on Fuzzy Systems 3(3), 336–356 (1995) (outstanding paper award IEEE TFS 1999)
[7] Kruse, R., Meyer, K.D.: Statistics With Vague Data. D. Reidel Publishing Company, Dordrecht (1987)
[8] Kruse, R., Siegel, P. (eds.): ECSQAU 1991 and ECSQARU 1991. LNCS, vol. 548. Springer, Heidelberg (1991)
[9] Kruse, R., Schwecke, E., Heinsohn, J.: Uncertainty and Vagueness in Knowledge Based Systems: Numerical Methods. Springer, New York (1991)
[10] Kruse, R., Gebhardt, J., Klawonn, F.: Foundations of Fuzzy Systems. John Wiley & Sons, Inc., Chichester (1994)
[11] Kruse, R., Gebhardt, J., Palm, R. (eds.): Fuzzy Systems in Computer Science. Vieweg-Verlag, Braunschweig (1994)
[12] Kruse, R., Borgelt, C., Held, P., Moewes, C., Steinbrecher, M.: Computational Intelligence. Textbooks in Computer Science. Springer, New York (to appear, 2012)
[13] Michels, K., Klawonn, F., Kruse, R., Nürnberger, A.: Fuzzy Control: Fundamentals, Stability and Design of Fuzzy Controllers. STUDFUZZ, vol. 200. Springer, Heidelberg (2006)
[14] Nauck, D., Klawonn, F., Kruse, R.: Foundations of Neuro-Fuzzy Systems. John Wiley & Sons, Ltd., Chichester (1997)
[15] Gabbay, D.M., Kruse, R., Nonnengart, A., Ohlbach, H.J.: FAPR 1997 and ECSQARU 1997. LNCS, vol. 1244. Springer, Heidelberg (1997)
[16] Schröder, M., Petersen, R., Klawonn, F., Kruse, R.: Two paradigms of automotive fuzzy logic applications. In: Jamshidi, M., Titli, A., Zadeh, L., Boverie, S. (eds.) Applications of Fuzzy Logic: Towards High Machine Intelligence Quotient Systems. Environmental and Intelligent Manufacturing Systems Series, vol. 9, pp. 153–174. Prentice-Hall, Inc, Upper Saddle River (1997)

Author Index