s c c h
software competence center
hagenberg

FUZZY.LOGIC.LAB.LINZ

Abstracts of the FLLL/SCCH Master and PhD Seminar

Room 010, Software Park Hagenberg

April 7, 2005

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

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

Program

Session 1 (Chair: Leila Muresan) 9:00–10:30

9:00 Mario Drobics:
FS-LiRT—An Inductive Learning Method for Creating Comprehensible Fuzzy Regression Trees

9:30 Holger Schöner:
Selected Research Topics in Data Analysis: Blind Source Separation and Missing Values

10:00 Eyke Hüllermeier:
Fuzzy-Methods in Knowledge-Discovery: On Frequency-Based Evaluation Measures

10:30 Coffee Break

Session 2 (Chair: Ulrich Bodenhofer) 11:00–12:00

11:00 Bettina Heise, Leila Muresan:
Some Aspects About PSF and Deconvolution

11:30 Leila Muresan, Bettina Heise:
Analysis of Spot Counting Algorithms in Fluorescence Microscopy Images

12:00 Lunch

Technical Report
SCCH–TR–0507

FS-LiRT—An Inductive Learning Method for Creating Comprehensible Fuzzy Regression Trees

Mario Drobics

Abstract — In this paper we present a novel approach to data-driven fuzzy modeling which aims to create highly accurate but also easily comprehensible models. This goal is obtained by defining a flexible but expressive language automatically from the data. This language is then used to inductively learn fuzzy regression trees from the data. Finally, a detailed comparison study on the performance of the proposed method and an outlook to future developments.

Key words — *fuzzy sets, inductive learning, decision trees*

Kplus
Kompetenzzentren-Programm

1 Introduction

Fuzzy logic based systems can be used to gain insights on a complex system for which no analytical model exists. For many complex technical applications the problem arises that no proper mathematical formulation can be found to describe the behavior of the according system. The only available information might be a set of measurements taken from the system. Then the goal is to find a function f that models the inherent connection between the input parameters (settings and measurements) and the goal parameter (final parameter of interest) that is hidden in the data.

To find such a function f, however, is not always the only objective. While statistical regression [DS81] or neural networks [MR86, RM86, Zur92] allow to solve such kinds of machine learning problems, they leave the resulting function f as a *black box*, i.e. a plain function whose internals are difficult or impossible to comprehend. In many practical applications, however, qualitative insights into the structures of f are desirable. For such tasks, *rule-based systems* are most appropriate. They easily allow qualitative insight, since the function f is represented by logical rules in a close-to-natural-language manner. In the following, assume that we are not necessarily interested in the full function f, but at least in significant bits of knowledge about f and their inherent structures. Rule based systems or decision trees [BFSO84, Qui93] have been profen to be easily comprehensible and are therefor ideal for this task of qualitative *and* qualitative analysis.

For the remaining, let us consider a data set $\mathcal{X}$ of K samples

$$\mathcal{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^K\}, \tag{1}$$

where each sample ($i = 1, \ldots, K$) has the same $(n+1)$-dimensional structure:

$$\begin{aligned} \mathbf{x}^i &= (x_1^i, \ldots, x_n^i, x_{n+1}^i) \\ &\in X_1 \times \cdots \times X_n \times X_{n+1} \end{aligned} \tag{2}$$

The first n dimensions/variables are the inputs; the last dimension/variable $n+1$ is the output under investigation. In the following, we refer to the r-th dimension ($r = 1, \ldots, n$) as *r-th input attribute*. The $n+1$-th dimension is called *goal attribute*. Ideally, the overall objective of this machine learning problem is then to find a function

$$f : X_1 \times \cdots \times X_n \longrightarrow X_{n+1} \tag{3}$$

such that the inherent connection between the input attributes and the goal attribute hidden in the data set $\mathcal{X}$ is modeled as well as possible. Therefore, such machine learning problems can be regarded as some kind of data fitting.

To be able to handle numeric attributes in rule-based models, it is indispensable to define a discrete set of predicates for these kinds of attributes. If this quantization is done by means of partitions into crisp sets (intervals) as in traditional machine learning, small variations (e.g. noise) can cause large changes in the classification quality and instable results. This entails the demand for admitting vagueness in the assignment of samples to predicates. Fuzzy sets [Zad65] perfectly solve this problem of artificial preciseness arising from sharp interval boundaries.

A second benefit of fuzzy logic systems like decision trees [Ada80, ZS96] or rule-based methods [TS85, BYTP03] is, that they create not only a computational but also an interpretable model for f. The resulting function helps the user to better understand the behavior of the system [CCHM03]. It turned out, however, that in many cases the simple application of methods for

creating interpretable, computational models from data is not sufficient. There is often the need for higher accuracy, by preserving the interpretability of the systems. Consequently, recently several approaches were developed to optimize given interpretable logic fuzzy systems [CCHM03]. These approaches, however, always focus on either interpretability *or* accuracy.

To overcome these limitations, we will compute semantically meaningful fuzzy sets a priori to the rule induction process, integrating user defined fuzzy predicates. We will then use these predicates for inductive learning of fuzzy decision trees to obtain comprehensible fuzzy models from data. To obtain models with higher accuracy a regularizes optimization technique can be applied to the whole model, afterward.

2 The Underlying Language

To define the underlying language for our fuzzy models, we have to consider the different types of input attributes that can occur. Basically, we can distinguish between three types of attributes:

Boolean categorical attributes: The domain X_i is an unstructured finite set of labels, for instance, types of car engines (gasoline, Diesel, hydrogen, electric) or classes of animals (birds, fish, mammals, etc.). The attribute values x_r^i are single elements of the label set X_i.

Fuzzy categorical attributes: There is again an unstructured finite set of labels, but with possible overlaps. Therefore, values of such kinds of variables may be fuzzy sets on this set of labels. For example, assume that we are given a finite set consisting of different grape varieties. Then blended wines (cuvees) cannot be assigned to single categories crisply.

Numerical attributes: The underlying domain X_i is the set of real numbers or a subset of these (e.g. an interval). The attribute values x_r^i are real numbers, e.g. pressures, temperatures, incomes, ratios, etc.

Note that Boolean categorical attributes are special cases of fuzzy categorical attributes, since any crisp label can be considered as a fuzzy set of labels, too.

Fuzzy predicates for categorical attributes, boolean or fuzzy, can be defined easily in a straight forward manner. Finding appropriate fuzzy predicates for numerical attributes, however, is often a subtle problem for which different approaches exist.

In our approach, we create the fuzzy sets based on the data set given by considering the semantics of the corresponding linguistic expressions automatically using a method called *CompFS* [Dro04]. By comprising also ordering based predicates we are able to define comprehensible, but still expressive predicates automatically [BB03].

3 Rule Induction

To create a decision or regression tree for a specific decision problem, *inductive learning* (i.e. learning from examples) is a widely used approach.

Using not only crisp but also fuzzy predicates, decision trees can be used to model vague decisions. Several approaches dealing with such fuzzy decision trees focus on the problem of

vague class memberships [MC93, Mar00, PF01, ZS96]. Viewing decision trees as a compressed representation of a (fuzzy) rule set, enables us to use decision trees not only for classification, but also for approximation of continuous output functions. Recent approaches in this direction try to create large trees, that solve the resulting optimization problem [BLM97, Jan98]. These solutions, however, can no longer be interpreted easily—which is usually one of the main advantages of regression trees over numerical optimization methods or artificial neural nets. Using pruning and back-fitting strategies can help to overcome this shortcoming [OW03]. All these approaches, however, tackle the problem of finding accurate but comprehensible models from an optimization point of view and do not pay attention to the underlying language used, nor are they capable of using a domain specific set of fuzzy predicates.

In our approach to inductive learning of fuzzy regression trees we pay special attention to comprehensibility, accepting a slightly lower performance compared to other approaches at this stage. This is achieved by using the general language defined in section 2 and by creating model, as compact as possible. The so obtained models is finally tuned with respect to predictive accuracy using a post optimization technique described in the subsequent section.

3.1 Fuzzy Regression Trees

A general regression tree consists of a *root node* with a number of *child nodes*. Each of these child nodes can either be a *leaf node* or the root node of a new subtree. If each non-leaf node has exactly two child nodes, the tree is called *binary*. We denote the set of all nodes with $\mathcal{N} = \{n^1, \ldots, n^N\}$, the set of all leaf nodes with $\mathcal{L} = \{n^{l_1}, \ldots, n^{l_L} \subset \mathcal{N}\}$ and the set of non-leaf nodes with $\mathcal{M} = \{n^{M_1}, \ldots, n^{M_N} \subset \mathcal{N}\}$ where we define the node n^1 to be the root node.

To each non-leaf node $n^i \in \mathcal{M}$, a predicate p^i is associated which is used to decide which of the child nodes to process next. For each non-leaf node $n^i \in \mathcal{M}$ the child nodes are denoted as n_1^i and n_2^i and we define that the left branch (n_1^i) is selected when the corresponding predicate p^i is fulfilled and the right one (n_2^i) otherwise. The uniquely determined path from the root node n^1 to a sub-node $n^j \in \mathcal{N}$ is called *complete branch* of the node and will be denoted as b^j. Each leaf node $n^j \in \mathcal{L}$ is associated with a constant value $c^j \in \mathbb{R}$ or a local model $c^j(\mathbf{x}), X \mapsto \mathbb{R}$.

In the following, we will restrict ourselves to binary regression trees. We overcome the main problem of binary trees—their increasing size for complex problems, by using a flexible underlying language, especially ordering-based predicates. This enables us to determine the ideal segmentation point automatically and to reduce the overall number of predicates involved.

3.2 Inductive Learning of Fuzzy Regression Trees—*FS-LiRT*

The basic idea behind *FS-LiRT* is to create a tree where the leaves approximate the desired goal function as good as possible. By associating numerical values (or functions) with the leave nodes, we finally obtain a Sugeno- or TSK-type controller. The method is called *FS-LiRT* (*F*uzzy *S*et based *Li*near *R*egression *T*rees) as in most cases linear models are associated with the leaf nodes.

We use the mean squared error measure which ensures that the model accuracy increases the larger the tree grows. The mean squared error for a given predicate P and a sample set $\mathcal{X}$ is

Algorithm 1 (FS-LiRT)

Input: goal attribute m
samples $X_{\text{cur}} = \{\mathbf{x}^1, \ldots, \mathbf{x}^K\}$
set of test predicates $\mathcal{P}$
Output: tree node N_{cur}

if stopping criterion is fulfilled
{
compute $c^{\text{cur}} = \{c_1^{\text{cur}}, \cdots, c_k^{\text{cur}}\}$
N_{cur} is leaf node with class assignment C_{cur}
}
else
{
find best predicate $P = \text{argmin}_{P \in \mathcal{P}} \text{MSE}_{\text{DT}}(P, X_{\text{cur}})$
compute new memberships for the left branch
$\mu_{X^{\oplus}}(\mathbf{x}^i) = t\big(\mu_{X_{\text{cur}}}(\mathbf{x}^i) \wedge P(\mathbf{x}^i)\big)$
compute left branch
$N^{\oplus} = \mathbf{FS-LiRT}(C, X^{\oplus}, \mathcal{P})$
compute new memberships for the right branch
$\mu_{X^{\ominus}}(\mathbf{x}^i) = t\big(\mu_{X_{\text{cur}}}(\mathbf{x}^i) \wedge \neg P(\mathbf{x}^i)\big)$
compute right branch
$N^{\ominus} = \mathbf{FS-LiRT}(C, X^{\ominus}, \mathcal{P})$
N_{cur} is parent node with children $N^{\oplus}$ and $N^{\ominus}$
}

computed according to:

$$\text{MSE}_{\text{DT}}(P, \mathcal{X}) = \frac{\sum_{\mathbf{x} \in \mathcal{X}} \mu_{\mathcal{X}}(\mathbf{x})(\tilde{z}_P(\mathbf{x}) - x_{n+1}))^2}{|\mathcal{X}|}, \tag{4}$$

$$\tilde{z}_P(\mathbf{x}) = t(P(\mathbf{x}))\bar{z}(\mathcal{X}|P) + t(\neg P(\mathbf{x}))\bar{z}(\mathcal{X}|\neg P), \tag{5}$$

where x_{n+1} is the desired goal value, $\tilde{z}_P(\mathbf{x})$ is an estimate of the output according to predicate P, and

$$\bar{z}(\mathcal{X}|P) = \frac{\sum_{\mathbf{x} \in \mathcal{X}} t(P(\mathbf{x}))x_{n+1}}{\sum_{\mathbf{x} \in \mathcal{X}} t(P(\mathbf{x}))}$$

is the average goal value with respect to the predicate P.

Outline: *Starting with a single root node a binary regression tree is grown in a top-down manner. The mean squared error is computed and the predicates are sorted with respect to their actual relevance. Then the most relevant predicate is chosen and associated with the tree node under investigation. This procedure is repeated recursively until a stopping condition is fulfilled.*

The leaf node output c^j for a leaf node l^j is defined as the weighted average of the $n+1$-th attribute (our goal attribute) according to:

$$c^j = \frac{\sum_{\mathbf{x} \in X} t(l^j(\mathbf{x}))x_{n+1}}{\sum_{\mathbf{x} \in X} t(l^j(\mathbf{x}))} \tag{6}$$

To achieve a more accurate approximation it is also possible to define the output c^j as a linear combination of the inputs. This can be achieved by solving the local least squares problem

$$\sum_{\mathbf{x}\in X}\Big(t(l^j(\mathbf{x}))\big(x_{n+1}-\alpha_{j0}-\sum_{i=1}^{n}\alpha_{ji}x_i\big)\Big)^2=\min_{\alpha_j}, \tag{7}$$

where $\alpha_j=(\alpha_{j0},\alpha_{j1},\ldots,\alpha_{j(n)})^T$ are the according linear weights.

Alternatively, we can define the output for a leaf node l^j also as those values which minimize the quantization error in the parent node. Suppose, that the node n^i is expanded by creating two child nodes $j_1=n_1^i$ and $j_2=n_2^i$. The optimal output values of these two nodes is then defined as the solution of the following minimization problem:

$$\frac{\sum_{\mathbf{x}\in\mathcal{X}}\left(t(l^j(\mathbf{x}))x_c-\big(t(l^{j_1}(\mathbf{x}))c^{j_1}+t(l^{j_2}(\mathbf{x}))c^{j_2}\big)\right)^2}{\sum_{\mathbf{x}\in\mathcal{X}}t(l^j(\mathbf{x}))}=\min_{c^{j_1},c^{j_2}} \tag{8}$$

The solution to this minimization problem can be computed efficiently using least squares optimization methods. The main benefit of this approach is, that it takes to a certain degree the structure of the tree into account, while the approaches in equation (6) and (7) only give a local solution. This results in a better coverage of the original output range. As, however, only two levels of the tree are considered, its influence decreases, the larger the tree grows.

The algorithm stops if any of the following stopping criteria is fulfilled (note, that if pruning is applied only the first stopping criteria is applied):

- No more samples: if the number of samples decreases under a certain threshold (Default: 10% of the orignal data).
- Minimum variance: if the variance of all samples in a node is below a given threshold (Default: 5% of the range of the goal attribute).
- Maximum depth reached: if the depth of the tree reaches a predefined maximum (Default: 10).
- No sufficient increase: if the relative increase with respect to the mean squared error

  $$\frac{\mathrm{MSE}(p^i,\mathcal{X}^i)-\mathrm{MSE}(p^j,\mathcal{X}^j)}{\mathrm{MSE}(p^i,\mathcal{X}^i)}$$

  (c^j being a child node of c^i) is below a given threshold (Default: 0.10).

Optionally, pruning can be applied to the tree generated by *FS-LiRT* to optimize the size of the tree. The goal is to achieve a good compromise between a models simplicity and its predictive accuracy, by removing irrelevant parts of the model. By pruning a tree, the new complexity of the model is automatically identified without the need to a priori establish thresholds for the stopping conditions which could be sensitive to problem specifics. Pruning also enhances the interpretability of a tree, a simpler tree being easier to interpret.

We use the same pruning technique as presented by [OW03]. They used a four step procedure, where first the inner nodes of the tree are sorted with respect to their sum-squared-error. Then a sequence of subtrees is generated by subsequently deleting the child nodes of the nodes in this sequence. Thirdly, the mean-absolute-error on a pruning data set is computed for each of these subtrees. Finally, the smallest tree within one standard error is selected.

3.3 Deduction with Fuzzy Regression Trees

To obtain real-valued output from fuzzy regression trees, the regression tree can be inferred directly by computing the output of a tree node $n^i(\mathbf{x})$ according to:

$$c^i(\mathbf{x}) = t(p^i(\mathbf{x}))c_1^i(\mathbf{x}) + t(\neg p^i(\mathbf{x}))c_2^i(\mathbf{x}) \tag{9}$$

This formula is evaluated recursively to obtain the output of the root node c^1.

Alternatively, we can transform the regression tree to a corresponding TSK fuzzy system. The transformation is performed according to:

$$\text{IF } n^j(\mathbf{x}) \text{ THEN } c^j(\mathbf{x}), \qquad \forall n^j \in \mathcal{L}. \tag{10}$$

The degree $n^j(\mathbf{x})$ to which a sample $\mathbf{x}$ belongs to the leaf node n^j is computed as the conjunction of the predicates on the corresponding complete branch b^j:

$$n^j(\mathbf{x}) = \bigwedge_{p^i \in b^j} b_i^j(p^i(\mathbf{x}))$$

The function $b_i^j(p^i)$ returns p^i when the left (true) branch and $\neg p^i$ when the right (false) branch of node n^i is part of b^j.

4 Examples

4.1 Two-Dimensional Example A

To illustrate the potential of the proposed method for fuzzy modeling, we tried to reconstruct the following function from data ($n = 2$, $X_1 = X_2 = [0, 100]$, $X_3 = [-100, 100]$):

$$f_{2D-A}(x_1, x_2) = x_2 \cdot \sin\left(\frac{2\pi x_1}{100}\right)$$

We selected $K = 1000$ random samples (x_1^i, x_2^i) from the range $X_1 \times X_2 = [0, 100]^2$. The final data set was constructed as

$$\mathcal{X} = \{\left(x_1^i, x_2^i, f_{2D-A}(x_1^i, x_2^i)\right) \mid i = 1, \ldots, K\}.$$

Six fuzzy sets with bell-shaped membership functions were created for the first input attribute x_1 and two for the second input attribute x_2 (see Figure 1).

FS-LiRT was executed to create a compact regression tree. In a second run, a larger tree was created and pruning was applied to remove unnecessary nodes. The pruned tree is shown in Fig. 2.

Parameter	Setting 1	Setting 2
Logic	Product	Product
$\text{supp}_{\min}$	0.05	0.01
$\text{stddev}_{\min}$	0.1	0.001
$\text{incr}_{\min}$	0.01	0.001
pruning	no	yes

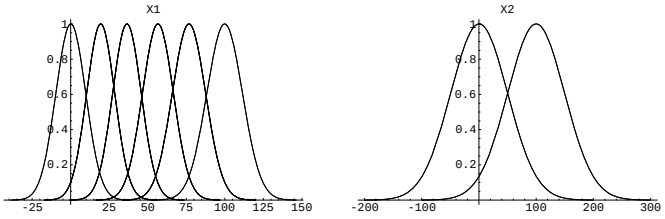

Figure 1: Fuzzy sets for the function approximation problem 2D-A

Finally, we compared the output of the resulting fuzzy controller with the original values. The results for the original and the pruned tree are shown below. While the original tree was much larger than the pruned tree, the pruned tree shows a almost equal performance. Figure 3 shows plots of the original function f_{2D-A} and the function defined by the resulting Sugeno system of the pruned tree.

Parameter	Setting 1	Setting 2
Tree size	22	7
Correlation	0.97	0.98
Average Error	8.18	8.15
Average Squared Error	97.87	100.44

4.2 Two-Dimensional Example B

Second, we tried to find a fuzzy controller to approximate a more complex two-dimensional function ($n = 2$, $X_1 = X_2 = [-1, 1]$, $X_3 = [0, 1]$):

$$d(x_1, x_2) = (x_1^2 + x_2^2)^{\frac{1}{2}}$$
$$f_{2D-B}(x_1, x_2) = \frac{\cos(5\pi d(x_1, x_2)}{1 + 10d(x_1, x_2)}$$

We selected $K = 1000$ random samples (x_1^i, x_2^i) from the range $X_1 \times X_2 = [-1, 1]^2$. The final data set was constructed as

$$\mathcal{X} = \{\big(x_1^i, x_2^i, f_{2D-B}(x_1^i, x_2^i)\big) \mid i = 1, \ldots, K\}.$$

Fifteen fuzzy sets with bell-shaped membership functions were created for the two input attributes x_1 and x_2. The domain X_3 of the goal attribute has been covered by six fuzzy sets with bell-shaped membership functions.

Finally, *FS-LiRT* was executed to create a regression tree. We ran *FS-LiRT* using the following parameter settings:

Parameter	Setting
Logic	Product
$\text{supp}_{\min}$	0.01
$\text{stddev}_{\min}$	0.01
$\text{incr}_{\min}$	0.01

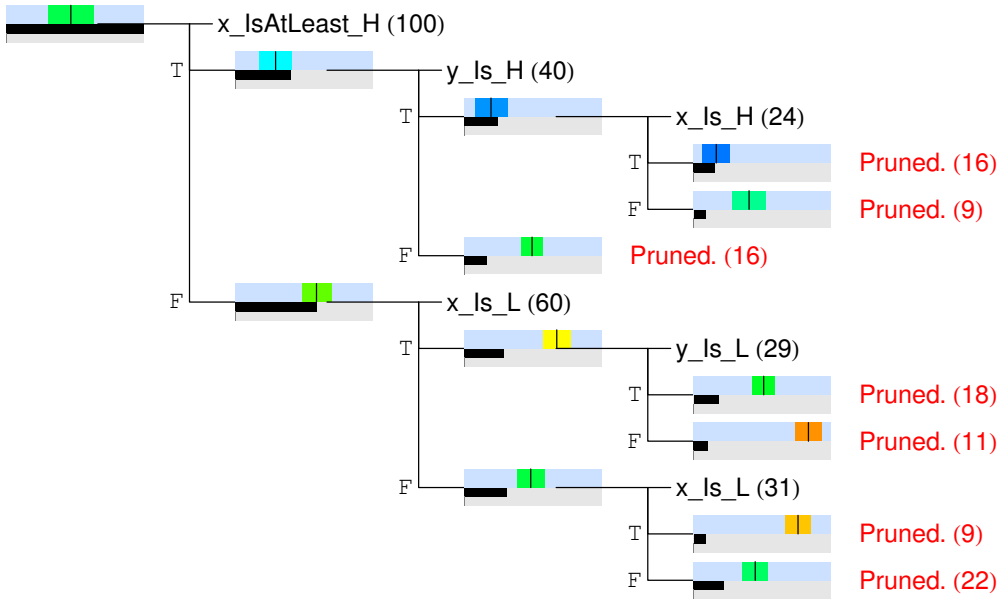

Figure 2: Regression tree constructed by FS-LiRT for the function approximation problem 2D-A after pruning

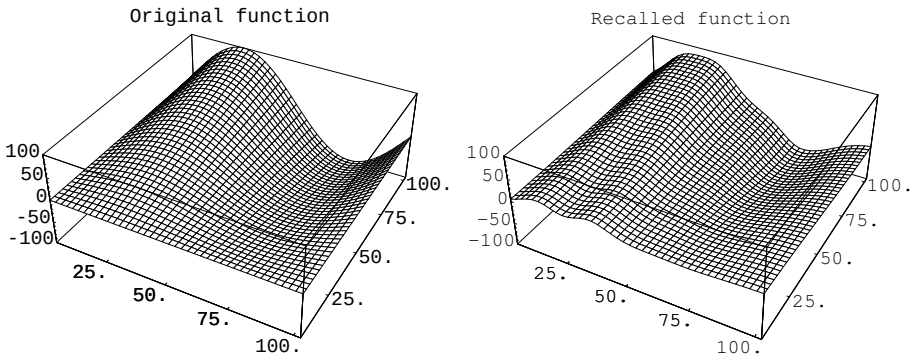

Figure 3: Test function f_{2D-A} (left) and the function defined by a Sugeno fuzzy system constructed by FS-LiRT (right)

In our experiments it showed, that pruning did not result in any performance improvements. This is most likely caused by the fact, that the function has a very local behavior, which can not be generalized easily.

Finally, we compared the output of the resulting fuzzy controller with the original values. The results for the original and the pruned tree are shown below. Figure 4.2 shows plots of the original function f_{2D-B} and the function defined by the resulting Sugeno system of the original tree.

Parameter	Result
Tree size	68
Correlation	0.9052
Average Error	0.092
Average Squared Error	0.013

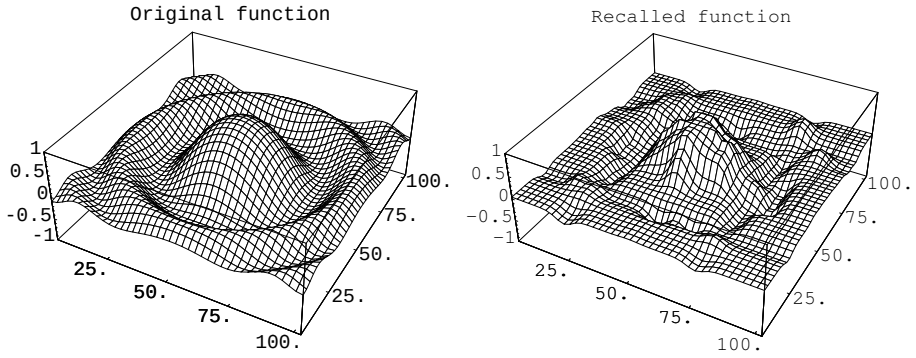

Figure 4: Original and recalled function for f_{2B} obtained using *FS-LiRT*

4.3 High-dimensional Data Set

To briefly illustrate the potential of the proposed method for fuzzy modeling, we tried to reconstruct the following widely used target function:

$$f_{6D}(x_1, x_2, x_3, x_4, x_5, x_6) = x_1 + x_2^{0.5} + x_3 x_4 + 2e^{2(x_5 - x_6)},$$

with $n = 2$, $X_1 = [1, 5]$, $X_2 = [1, 5]$, $X_3 = [0, 4]$, $X_4 = [0, 0.6]$, $X_5 = [0, 1]$, and $X_6 = [0, 1.2]$. We selected $K = 1000$ samples (x_1^i, x_2^i) from $X_1 \times X_2 \times X_3 \times X_4 \times X_5 \times X_6$ for training and $K = 2000$ random samples for validation.

Finally *FS-LiRT* was executed to create a regression tree. We ran *FS-LiRT* using the following parameter settings:

Parameter	Setting 1	Setting 2
Logic	Product	Product
$\mathrm{supp}_{\min}$	0.025	0.025
$\mathrm{stddev}_{\min}$	0.1	0.001
$\mathrm{incr}_{\min}$	0.01	0.001
pruning	no	yes

Then we compared the output of the resulting fuzzy controller with the original values. Figure 5 shows a comparison of the original function f_{6D} and the function defined by the resulting Sugeno system using the second parameter setting.

Parameter	Setting 1	Setting 2
Tree size	62	52
Correlation	0.955	0.953
Average Error	0.6	0.617
Average Squared Error	0.65	0.676

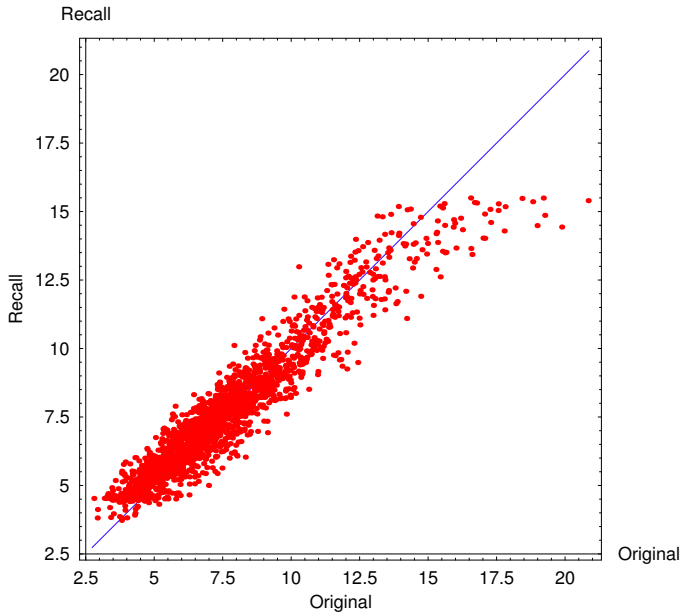

Figure 5: Input-Recall plot for f_{2B} obtained using *FS-LiRT*

4.4 Comparison of Results

To illustrate, how the rule induction methods can be used for numerical prediction, we have applied them on three regression problems from the UCI repository and on three artificial two-dimensional regression problem. Again, we used 10-fold cross validation and computed the average correlation coefficient σ. We compared the obtained results with two of our own methods, *FS-ID3* [DB02] and *FS-FOIL* [DBK02], as well as three methods from the WEKA toolkit [WF00], namely *Linear-Regression*, *M5-Prime* [Qui92], and *M5-Rules*. The first method creates a simple linear regression model to solve the regression learning problem. The latter two methods, *M5-Prime* and *M5-Rules* generate decision trees or decision rules to do so. We ran these methods using only constant output values, but with smoothing enabled using the Weka Toolbox 3-4.

First, we compared the average correlation coefficient between the original output and the predicted values for the test data. The results are shown in Figure 6. We can see, that for the data sets from the UCI repository our methods performed equally well as the other methods. We can, however, also observe, that the results obtained using simple linear regression have the same performance, too. This indicates, that these problems are almost linear and complex methods do not provide any additional benefit at all. For the complex two dimensional problems where oscillations are involved, the more complex methods outperform linear regression easily. In these cases our methods perform also much better then *M5-Prime* and *M5-Rules*. This is mainly caused by the fact, that using exponential type fuzzy sets enables us to create smooth transitions between the different rules.

	AUTO-MPG	HOUSING	SERVO	2D-A	2D-B	6D
CreateID3	0.893332	0.863603	0.873331	0.91616	0.923069	0.949423
CreateLIRT	0.905819	0.874305	0.940749	0.953215	0.850051	0.959669
CreateFOIL	0.564103	0.836021	0.756518	0.892246	-0.144268	0.777745
LinearRegression	0.923931	0.848237	0.848528	0.6455	0	0.9122
M5Rules	0.881499	0.816674	0.861871	0.8722	0.8427	0.8797
M5P	0.909242	0.883161	0.899288	0.8563	0.8641	0.8995

Figure 6: Comparison of average correlation for different regression problems

Finally, we compared the corresponding size of the models learned. The results are shown in Figure 7. Again, we can see that for the UCI data sets the size of the resulting models is almost equal. For the artificial data sets, however, the size of the models obtained using *FS-LiRT* are more compact than those obtained using the other methods. Again, the reason for this behavior is the usage of exponential type fuzzy sets. Interestingly, the size of the rule bases is in general smaller than of the regression trees. Detailed evaluation results are shown in Table 1.

	AUTO-MPG	HOUSING	SERVO	2D-A	2D-B	6D
CreateID3	10	17	18	6	79	39
CreateLIRT	15	24	11	13	53	69
CreateFOIL	4	8	8	3	1	8
LinearRegression	0	0	0	0	0	0
M5Rules	12.3	13.1	5.1	85.6	29	51.1
M5P	10.6	17.7	7.95	127.25	106.25	84.15

Figure 7: Comparison of average model size for different regression problems

Data Set	Algorithm	Correlation	MSE	MAE	Std.Dev.	Model Size
AUTO-MPG	CreateID3	0.893332	12.7866	2.58567	3.57808	10
AUTO-MPG	CreateLIRT	0.905819	10.9482	2.43566	3.31243	15
AUTO-MPG	CreateFOIL	0.564103	60.7159	5.25665	6.47952	4
HOUSING	CreateID3	0.863603	24.1744	3.63304	4.75783	17
HOUSING	CreateLIRT	0.874305	19.9721	3.13571	4.46946	24
HOUSING	CreateFOIL	0.836021	26.1163	3.49575	5.08848	8
SERVO	CreateID3	0.873331	0.579161	0.398723	0.760041	18
SERVO	CreateLIRT	0.940749	0.281946	0.335655	0.532581	11
SERVO	CreateFOIL	0.756518	1.05635	0.502508	1.03063	8
2D-A	CreateID3	0.91616	309.612	13.8261	17.5936	6
2D-A	CreateLIRT	0.953215	175.52	10.5095	13.138	13
2D-A	CreateFOIL	0.892246	845.961	22.0571	29.0934	3
2D-B	CreateID3	0.923069	0.0186727	0.112004	0.135186	79
2D-B	CreateLIRT	0.850051	0.0227944	0.125351	0.151023	53
2D-B	CreateFOIL	-0.144268	0.107258	0.266526	0.289967	1
6D	CreateID3	0.949423	0.834273	0.71752	0.896518	39
6D	CreateLIRT	0.959669	0.612081	0.61876	0.782518	69
6D	CreateFOIL	0.777745	4.96319	1.61919	2.08278	8

Table 1: Detailed results for different regression problems

5 Outlook

While in this work we have only considered locally approximating functions, it is necessary to consider the complete fuzzy model to obtain a globally optimal solution. We may then optimize the shape of the underlying fuzzy sets, the local output functions, or both. While modifying the underlying fuzzy sets together with the output functions will result in more accurate models it turns out, that defining the output functions in a globally optimal way leads to very good results and is computationally much less expansive.

Acknowledgements

This work has been done in the framework of the K*plus* Competence Center Program which gratefully acknowledges support by the Austrian Government, the State of Upper Austria, and the Johannes Kepler University Linz in the framework of the K plus Competence Center Program.

References

[Ada80] J. M. Adamo. Fuzzy decision trees. *Fuzzy Sets and Systems*, 4:207–219, 1980.

[BB03] U. Bodenhofer and P. Bauer. A formal model of interpretability of linguistic variables. In J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, editors, *Interpretability Issues in Fuzzy Modeling*, volume 128 of *Studies in Fuzziness and Soft Computing*, pages 524–545. Springer, Berlin, 2003.

[BFSO84] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, editors. *Classification and Regression Trees*. CRC Press, 1984.

[BLM97] J. F. Baldwin, J. Lawry, and T. P. Martin. A mass assignment based ID3 algorithm for decision tree induction. *Int. J. Intell. Syst.*, 12:523–552, 1997.

[BYTP03] P. Baranyi, Y. Yam, D. Tikk, and R. J. Patton. Trade-off between approximation accuracy and complexity: TS controller design via HOSVD based complexity minimization. In J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, editors, *Interpretability Issues in Fuzzy Modeling*, volume 128 of *Studies in Fuzziness and Soft Computing*, pages 249–277. Springer, Berlin, 2003.

[CCHM03] J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, editors. *Interpretability Issues in Fuzzy Modeling*, volume 128 of *Studies in Fuzziness and Soft Computing*. Springer, Berlin, 2003.

[DB02] M. Drobics and U. Bodenhofer. Fuzzy modeling with decision trees. In *Proc. 2002 IEEE Int. Conf. on Systems, Man and Cybernetics*, volume 4, Hammamet, Tunisia, October 2002.

[DBK02] M. Drobics, U. Bodenhofer, and E. P. Klement. FS-FOIL: An inductive learning method for extracting interpretable fuzzy descriptions. *Internat. J. Approx. Reason.*, 2002. (to appear).

[Dro04] M. Drobics. Choosing the best predicates for data-driven fuzzy modeling. In *Proc. 13th IEEE Int. Conf. on Fuzzy Systems*, pages 245–249, Budapest, July 2004.

[DS81] N. R. Draper and H. Smith. *Applied Regression Analysis*. John Wiley & Sons, New York, 1981.

[Jan98] C. Z. Janikow. Fuzzy decision trees: Issues and methods. *IEEE Trans. Syst. Man Cybern. B*, 28(1):1–14, 1998.

[Mar00] C. Marsala. Fuzzy decision trees to help flexible querying. *Kybernetika*, 36(6):689–705, 2000.

[MC93] P. E. Maher and D. S. Clair. Uncertain reasoning in an ID3 machine learning framework. In *Proc. 2nd IEEE Int. Conf. on Fuzzy Systems*, pages 7–12, San Francisco, CA, 1993.

[MR86] J. L. McClelland and D. E. Rumelhart, editors. *Parallel Distributed Processing—Exploration in the Microstructures of Cognition, Volume II: Psychological and Biological Models*. MIT Press, Cambridge, MA, 1986.

[OW03] C. Olaru and L. Wehenkel. A complete fuzzy decision tree technique. *Fuzzy Sets and Systems*, 138(2):221–254, 2003.

[PF01] Y. Peng and P. A. Flach. Soft discretization to enhance the continuous decision tree induction. In *Proc. ECML/PKDD01 Workshop Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, pages 109–118, 2001.

[Qui92] J. R. Quinlan. Learning with Continuous Classes. In *Proc. 5th Austr. Joint Conf. on Artificial Intelligence*, pages 343–348, 1992.

[Qui93] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[RM86] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing—Exploration in the Microstructures of Cognition, Volume I: Foundations*. MIT Press, Cambridge, MA, 1986.

[TS85] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. Syst. Man Cybern.*, 15(1):116–132, 1985.

[WF00] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, 2000.

[Zad65] L. A. Zadeh. Fuzzy sets. *Inf. Control*, 8:338–353, 1965.

[ZS96] J. Zeidler and M. Schlosser. Continuous valued attributes in fuzzy decision trees. In *Proc. 8th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 395–400, 1996.

[Zur92] J. M. Zurada. *Introduction to Artificial Neural Networks*. West Publishing, St. Paul, 1992.

Selected Research Topics in Data Analysis: Blind Source Separation and Missing Values

Holger Schöner
Software Competence Center Hagenberg
email *holger.schoener@scch.at*

Abstract — The following is a handout accompanying the presentation of selected PhD research topics in the First Joint FLLL/SCCH Master and PhD Seminar, Summer Term 2005. All presented topics share their background in Machine Learning and Data Analysis. First, the development and application of a decorrelation procedure to Optical Imaging of Intrinsic Signals is presented. Secondly, an extended Maximum Entropy Model is presented, and its application to heterogeneous data and analysis of data with missing values is discussed. The third and final topic is an extension of the Gauss kernel for Support Vector Machines, to allow analyses of incomplete data sets.

Key words — *Blind source separation, optical imaging, support vector machines, approximate maximum entropy modelling, missing values.*

PhD-Thesis

1 Introduction

The topics presented in the following were developed during my PhD research at the Berlin University of Technology, in the Neural Information Processing group with Prof. Obermayer, between 1999 and 2004.

The following sections will concentrate on conveying the general picture of the presented algorithms and approaches, without mathematical detail. If you would like to develop a deeper understanding, my thesis, with a detailed account of this research and further references, is available (also online) as [Schöner, 2004].

2 Blind Source Separation for Optical Imaging

2.1 Optical Imaging

The term Optical Imaging (OI) summarizes a collection of methods for acquisition and processing of brain image data. The goal is to support investigations and diagnostics of the functional architecture of human and animal brains; this summary concentrates on optical imaging of the visual cortex. For a detailed introduction see [Bonhoeffer and Grinvald, 1996].

The technique of *Optical Imaging of Intrinsic Signals* uses very slight variations of light reflectance and absorption properties of the neural tissue[1] during neural activity to build functional maps of the brain. In these, brain regions are color coded for preferred stimuli. As an example, in ocular dominance maps, regions of the visual cortex preferring input from the left eye are coded in a different color than regions preferring input from the right eye.

Image Sequences are taken by CCD or video cameras during presentation of different stimuli from the exposed visual cortex under monochromatic illumination (500 – 800 nm wavelength). The signal of interest is called mapping signal, and it is closely related to local activity of neuron populations. This signal is overlaid by other, very strong signals. Among these are pulsing blood vessels, global activity (also stimulus related, but on a coarser spatial resolution), further, not stimulus related signals, and different sources of noise. As the mapping signal usually constitutes only about 0.1% of total image intensity, its extraction is quite an interesting problem.

Conventional techniques, mainly image summation in conjunction with high- and low-pass filtering and differencing the imagas acquired for different stimuli, use some questionable assumptions, which led us to evaluate another approach, *Blind Source Separation*, presented in the following.

2.2 Blind Source Separation

Blind Source Separation (BSS) algorithms allow to extract source signals from several mixtures of several sources, if certain assumptions are met. One of these assumptions is that of a stationary linear mixing process, i.e. the mixing can be represented by a multiplication of a mixing matrix with source vectors. Usually, the number of observed mixtures has to equal the number of extracted

[1] In contrast to *Optical Imaging using Voltage Sensitive Dyes*. The latter leads to better detectable signals, but is not usable for certain experiments because of the toxicity of the dyes.

sources. And as a third prerequisite, the original sources are assumed to be statistically independent, or at least uncorrelated. These algorithms work by estimating a demixing matrix, which optimizes some kind of measure of statistical independence of the resulting estimated sources.

Well known for this kind of problems are ICA (*Independent Component Analysis*) algorithms, but we found an adaptation of a multiple decorrelation procedure to be better suited for optical imaging data, because it explicitly uses the fact, that the recorded images are spatially smooth. This adaptation is based on [Molgedey and Schuster, 1994], which diagonalizes correlation matrices of the time series, for two different lags of the correlation. Our adaptations are two-fold: First, we apply the decorrelation not to time series, but to pixels of the recorded images (decorrelating them for different spatial shifts); secondly, we modified the algorithm to allow simultaneous diagonalization of several correlation matrices, which makes the algorithm less moise sensitive and eases the selection of shifts.

For analysis of optical imaging data, the sources are assumed to be prototype images for different signal components, while the mixtures are the different recorded image frames. Each source can be present to a different degree in the observed mixtures, depending on its time course. The BSS algorithm should then decorrelate the observations, and the mapping signal is chosen among the resulting estimated sources by visual inspection or plausibility of time course.[2]

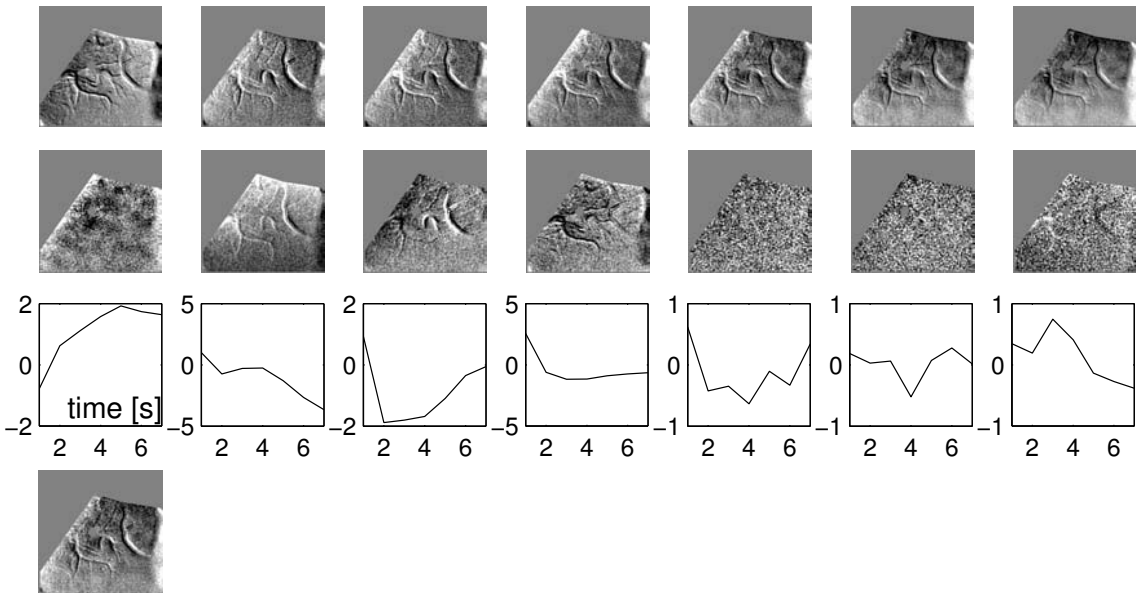

Figure 1: Application of Blind Source Separation to Optical Imaging data (Ocular Dominance experiment). First row shows raw preprocessed data; second row contains the sources estimated by BSS; third row are the time courses of the estimated sources; fourth row is the mapping signal extracted using the conventional summation algorithm.

[2]The mapping signal cannot be present before stimulus onset, and some knowledge about plausible rise and decay times is available.

2.3 Application

Figure 1 shows the application of the previously described algorithm to a real optical imaging data set. The mapping signal estimate (first image in second row) is well separated from the other sources, while it is not discernible in the original data (first row), even though that is already preprocessed by temporal binning and low-pass filtering. Using the time courses of the estimated sources, it is easy to select the right source as mapping signal among those found by BSS. The conventional summation technique cannot separate the mapping signal from the blood vessel artifacts.

In conclusion, the presented approach can be very usefully applied to optical imaging. The assumptions on a linear mixing process seem to be approximately fulfilled, as is also the case for the decorrelated sources assumption (if decorrelation is performed in the spatial instead of the temporal domain). A problem with the presented algorithm is, that it breaks down, if there are movement artifacts in the data, e.g. moving vessels. In that case, the linear mixing assumption does not hold any more, the algorithm estimates a lot of very strong "virtual" sources contain the blood vessels, which strongly dominate the mapping signal.

3 Approximate Maximum Entropy

The *Approximate Maximum Entropy* model (AME), [Yan and Miller, 2000], seemed to provide an elegant way to deal with incomplete data (missing values in a data matrix) and heterogeneous features (categorical features and values on differing scales) in the context of classification. This was the motivation for my research leading to an extension and evaluation of this model.

3.1 Basic Idea

The basic idea behind this model is to learn classification tasks by only taking explicitly stated knowledge into account, while not making any unwarranted assumption beyond that. The realization makes use of maximum entropy techniques introduced by [Jaynes, 1957]. The learning takes place as maximization of the entropy of the joint pdf (probability density function) of input features and classification goal, while the knowledge about the problem is stated explicitly as constraints to this optimization. This optimization has traditionally been intractable for realistic data set sizes, but some approximations presented in [Yan and Miller, 2000] now allow to perform this for data sizes of several thousands of examples with a few hundred features. Instead of describing the mathematics of this in detail, the following concentrates on an intuitive understanding of the way this model works.

Figure 2 gives an illustration of an AME model. It learns a representation of the joint pdf of features and target(s), and is very similar in structure and properties to *Boltzman machines*. In contrast to e.g. *multilayer perceptrons*, it has a stochastic nature. That means, every node assumes values according to a conditional probability distribution for that node, where the conditioing takes place over the values of all nodes, which are connected by weights to that node. To make a prediction using this model, the input nodes are clamped to the input values observed, followed by the computation of the probabilities for the output/target value. In this regard, the AME model is also similar to *Bayesian Networks* (BN). Among the differences to these are the kind of dependencies

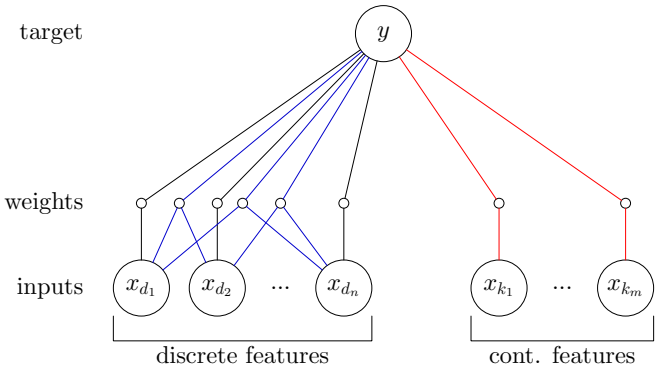

Figure 2: Graphical representation of an AME model.

between the nodes (statistics of all orders for BN, only selected orders for AME), which is also evident by the use of weights between nodes in the AME model. The concentration on some dependencies, on the other hand, allows to take into acocunt dependencies between many features, even if these are only weak. Another advantage is, that dependencies of lower order can usually be estimated more reliably empirically than those of higher order.

The weights are learned as described using constrained maximization of the joint entropy. Every weight in the model corresponds to one constraint, and its strength is adapted to exactly enforce that constraint. Possible constraints include the fixation of occurence frequencies of values of two or more features to those frequencies observed in a data set, for discrete features. Obvious constraints for continuous features are the consistency of empirical and model moments. The flexibility of constraints which can used allows very flexible integration of features with heterogeneous structure, as well as incorporation of prior knowledge. Furthermore, by learning all relevant weights, it is possible to perform general inference, where not one feature is explicitely marked as a target, but where potentially every feature may be predicted, given (a subset of) the others.

3.2 Missing Values

The AME model allows several ways of missing values treatment. First, it provides a representation of the joint pdf of all features, which can be used to marginalize over the possible values of missing features. Alternatively, the general inference mechanism can be used to predict some or all of the missing values. Both of these methods should only be used, if not too many values are missing (for performance reasons), and if the fact, that a value is missing, is not informative about the feature to predict.

A third possibility, which is appropriate for informatively missing values, is the introduction of an extra value for each feature, which is substituted for each missing value. This conserves knowledge about the fact, that these values are missing, and the ability to work with discrete data does not introduce a questionable interpretation of such values, which might arise, if this is done for other models as well, e.g. support vector machines or multilayer perceptrons.

Another useful method is applicable, if there are more uninformatively missing values, than can be treated by the first two methods, when these occur only during application of a learned model. Given a complete trained model and the training data, it is possible to derive the model appropriate for all subspaces of known features (subspace classificators).

3.3 Conclusions

Evaluation of the presented model on artificial and customer data sets showed, that it can provide advantages especially in those cases, for which I originally looked for an alternative to support vector machines and multilayer perceptrons: if the features are very heterogeneous, and if the flexibility in missing values treatment is necessary. It is less applicable to small data sets (few examples), because the approximations used are not valid in this case.

Another positive property is the interpretability of AME models: the relation of the weights to certain features allows the identification of the kind of dependencies in the data.

4 Support Vector Machines Extension

Support Vector Machines (SVMs) are a currently very successful and actively researched machine learning technique, [Schölkopf and Smola, 2002]. They were developed with the explicit goal of providing good generalization abilities when machines trained on one data set are applied to new data. Unfortunately, there does not exist a technique to allow them the direct treatment of missing values. Data sets with missing values have to be preprocessed such that only complete data is used for training.

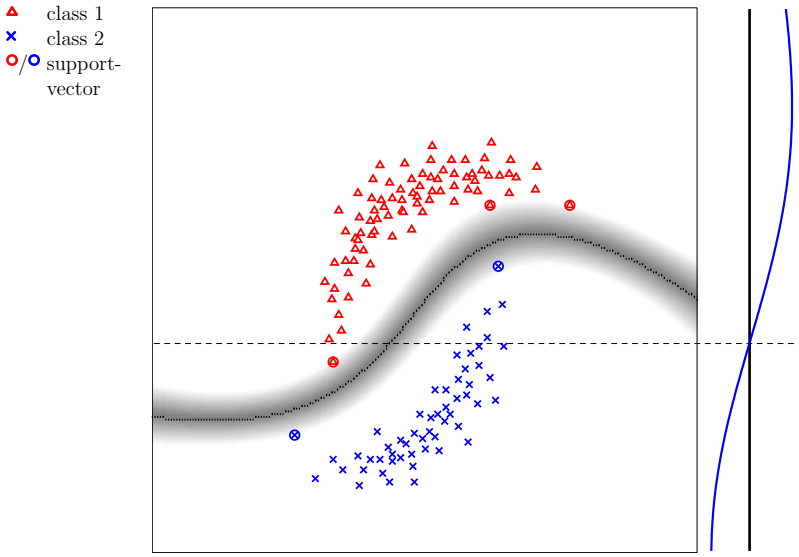

Figure 3: A simple classification problem with two classes (triangles red / crosses blue). The classification boundary learned by the SVM is shown in black, the margin in shades of gray. On the right side, the decision function of the subspace SVM for the vertical axis is shown, and its classification boundary in the original space is indicated by dashes.

Here again, the subspace classificator idea proves sensible. It would be desirable to have the ability to derive a SVM for a subspace of known features from a complete-space SVM. Unfortunately this is not possible to do exactly, because the support vectors used by the machines may be different. Training a SVM for all possible subspaces on the other hand leads to a combinatorial explosion. Anyway, using the support vectors of the complete-space machine also for the

subspace-machines provides an approximation for SVMs using Gauss-kernels, which is shown to be useful for certain kinds of data sets in the following.

In figure 3, this is illustrated for a simple toy example. If the complete space SVM is projected onto the vertical axis, the dashed classification boundary results (in the original space). Although this is not optimal (it should probably be placed a little bit higher to misclassify the least number of examples on new data), it provides a reasonable approximation.

Figure 4 on the other hand shows the classification accuracy of SVMs with Gauss kernels using different missing values handling techniques. Training (horizontal axis) took place on a complete training set, using the presented subspace classifier idea, using different imputation (filling in) strategies (Nearest Neighbor Imputation NN, NN where the nearest neighbor is chosen only from the same class, mean imputation), and finally using extra binary features for each feature, which indicate, whether the corresponding original feature is missing (this was filled in using mean imputation). Fewer methods were used for testing (see legend), because the "NN Same Class" and "no MV" do not make sense or cannot be applied on test data.

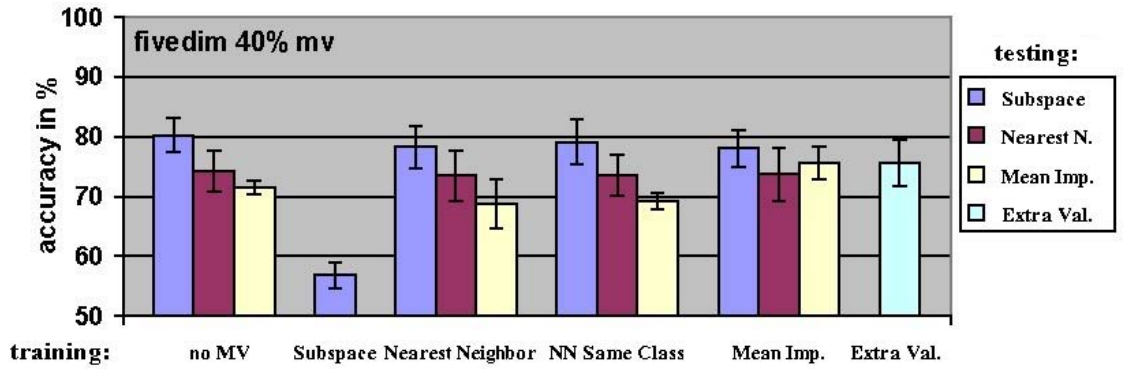

Figure 4: Test set accuracy for different combinations of missing values handling methods during training and testing. Error bars are at one standard deviation for 5 runs, but do not truly reflect variation in the algorithm but that due to different training/test set compositions.

Although the "Subspace" idea is clearly not suitable during training, it provides the best results during testing (especially when combined with "NN Same Class" or "Nearest Neighbor" for training). These advantages are prominent especially, when the data consists of several blobs, which are arranged non-trivially, i.e. which cannot be linearly separated. "Subspace" does not work well during training, because in that case usually several examples with missing values become support vectors, and their influence dominates the position of the classification boundary.

References

T. Bonhoeffer and A. Grinvald. Optical imaging based on intrinsic signals: The methodology. In A. Toga and J. C. Maziotta, editors, *Brain mapping: The methods*, pages 55–97. Academic Press Inc., San Diego, CA, USA, 1996.

E.T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4), May 1957.

L. Molgedey and H.G. Schuster. Separation of a mixture of independent signals using time delayed correlations. *Phys. Rev. Lett.*, 72:3634–3637, 1994.

B. Schölkopf and A. Smola. *Learning with Kernels*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, USA, 2002.

H. Schöner. *Working with Real-World Datasets – Preprocessing and prediction with large incomplete and heterogeneous datasets*. PhD thesis, Berlin University of Technology, Faculty of Electrical Engineering and Computer Science, Institute for Software Engineering and Theoretical Computer Science, Neural Information Processing Group, Franklinstrasse 28/29, D-10587 Berlin, 2004. URL `http://edocs.tu-berlin.de/diss/2004/schoener_holger.pdf`.

L. Yan and D.J. Miller. General statistical inference for discrete and mixed spaces by an approximate application of the maximum entropy principle. *IEEE Trans. on Neural Networks*, 11(3): 558–573, May 2000.

Fuzzy-Methods in Knowledge-Discovery: On Frequency-Based Evaluation Measures

Eyke Hüllermeier

Institut für Technische und Betriebliche Informationssysteme (ITI)
Otto-von-Guericke-Universität Magdeburg
D-39106 Magdeburg, Germany
huellerm@iti.cs.uni-magdeburg.de

In order to allow for the analysis of data sets including numerical attributes, several generalizations of association rule mining based on fuzzy sets have been proposed in the literature. While the formal specification of fuzzy associations is more or less straightforward, the assessment of such rules by means of appropriate quality measures is less obvious. Particularly, it assumes an understanding of the semantic meaning of a fuzzy rule. This aspect has been ignored by most existing proposals, which must therefore be considered as ad-hoc to some extent. In this paper, we develop a systematic approach to the assessment of fuzzy association rules. To this end, we proceed from the idea of partitioning the data stored in a database into examples of a given rule, counterexamples, and irrelevant data. Evaluation measures are then derived from the cardinalities of the corresponding subsets. The problem of finding a proper partition has a rather obvious solution for standard association rules but becomes less trivial in the fuzzy case. Our results not only provide a sound justification for commonly used measures but also suggest a means for constructing meaningful alternatives.

Some aspects about PSF and deconvolution

Bettina Heise, Leila Muresan

Background:

In microscopy imaging two important quality measures are the values of resolution *R* and contrast *C*. The resolution limit is determined by the Abbe relation,

$$R_{x,y} = \lambda /(2*NA), \quad R_z = n \lambda /NA^2$$

or the Rayleigh criterion. On the other hand high numerical apertures *NA* cause a decrease in contrast of the images. So especially wide field microscopy images are characterized to be always slightly blurred and seem to be covered with haze. Methods to improve this can be done in optical way (e.g. by confocal microscopy) or mathematically by deconvolution methods.

Describing the imaging process of a microscope by the convolution process

$$g(x, y) = f(x, y) * h(x, y) + n(x, y),$$

where $g(x, y)$ denotes the measured image, $f(x, y)$ is the 'true' image, $h(x, y)$ is the impulse response function of the system, which is called Point Spread Function (PSF) in optics. The Fourier transform of the PSF is often important and determined as Optical Transfer Function (OTF). The noise $n(x, y)$ can mostly be assumed as Gaussian or Poisson distributed. The aim of the deconvolution is to reconstruct the original 'true' image. There are several different methods of deconvolution, which can be distinguished, depending on known or unknown PSF, but also in iterative or direct linear methods, methods for one ore more image slices (3D or 2D-models).

Main types of deconvolution:

-Linear methods: Inverse Filter, PseudoInverse Filter, Wiener Filter, Nearest neighbour deconvolution, [Agard]

-Iterative constrained deconvolution: The initially estimated image f_0 is reblurred by PSF h, the differences to the measured image g are take into account for improvement of the estimation of f_1 , whereupon the iteration starts again…

-Maximum Likelihood deconvolution: For given PSF the solution is the function f_M which has the highest probability to be correct. ML-approaches have proved to give the best image quality. Mostly they are based on diffraction theory and Poisson statistics [Holmes].

For the above mentioned methods knowledge about the PSF is necessary. This can be done for instance by previous measurements with microbeads of known size or theoretical calculations.

-Blind deconvolution or combinations of them (Blind maximum likelihood deconvolution, [Bhattacharyya]).

In the case of blind deconvolution we have no exact knowledge about the PSF. This approach is used to estimate simultaneously the 'true' image and the PSF. Because the problem is underdetermined there must be included additional assumptions for uniqueness, for instance constraint to positive values of $f(x, y)$ or excluding delta function as solution. In practice it is done by assuming a physically based model of PSF with a set of unknown parameters. These parameters can be estimated by maximizing the log-likelihood- functional according to the imaging process.

Matlab-Realisation:

Several deconvolution methods are also pre-implemented in the Matlab Image Processing toolbox as Wiener filter, Lucy-Richardson or blind deconvolution. We tested them for our single protein images.

After applying a spot selection algorithm, the best (brightest) p% spots are selected. In our case p = 30%. If overlapping spots might occur, we can select the spots between the $[p_1, p_2]$ percentiles of the result set.

The selected spots are pixel-wise averaged and the resulting smoothed, denoised PSF is used in the Lucy-Richardson deconvolution. Some results are shown in Fig.1 and 2. Note the artifacts of the background in the case of blind deconvolution, compared to the Lucy-Richardson method. The problems are a good choice of the parameters and the time complexity and speed of the deconvolution in Matlab.

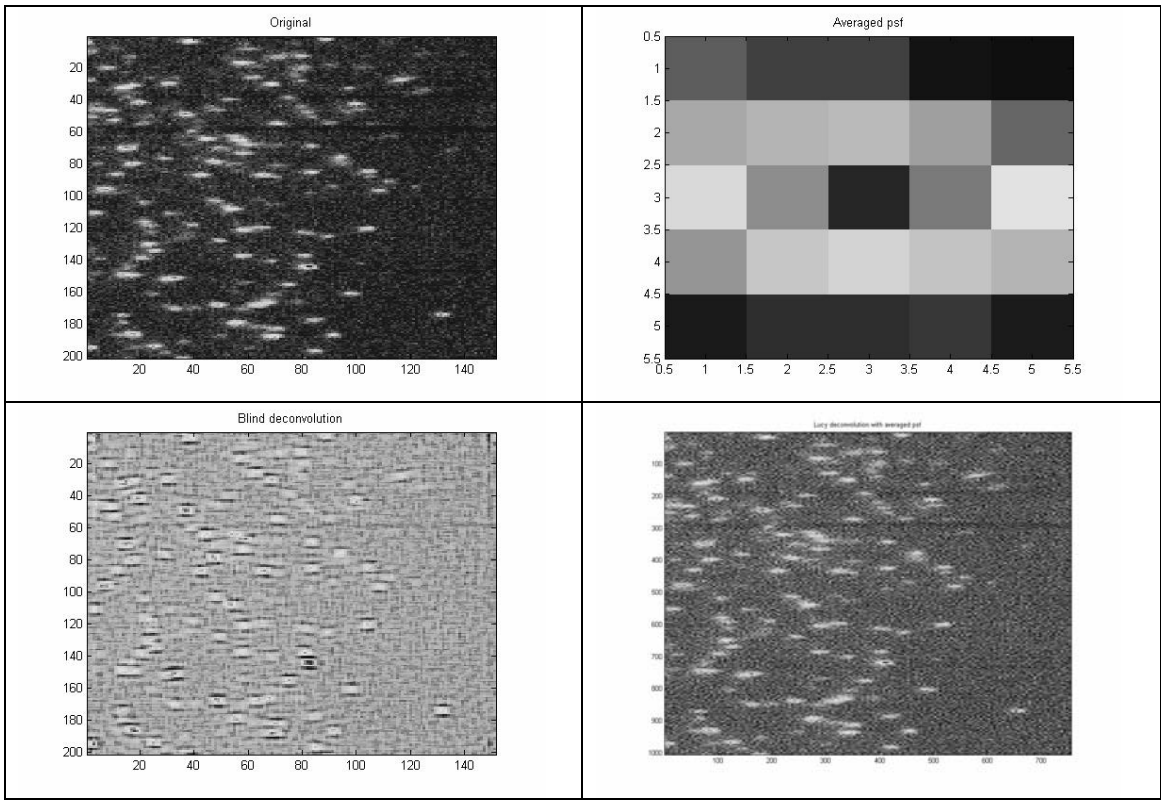

Fig. 1: Deconvolution of Microarray image spots

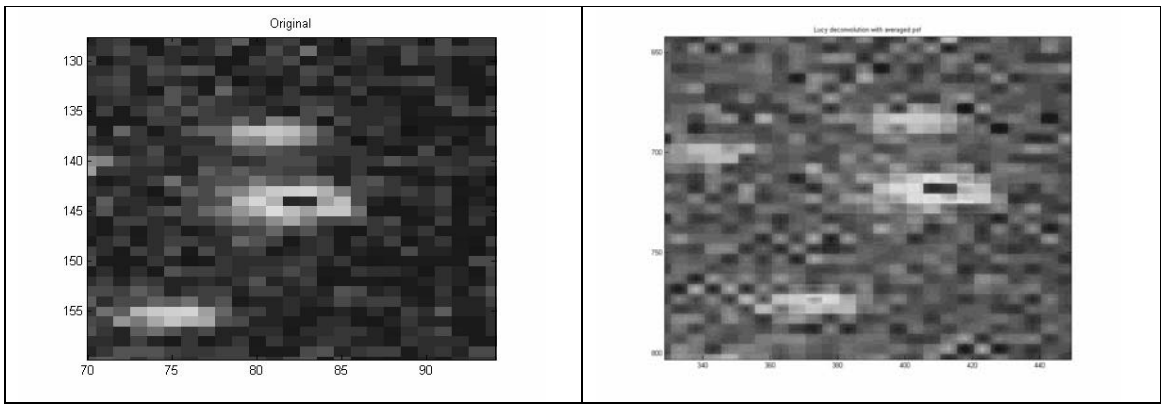

Fig.2: Detail of deconvolution by Lucy-Richardson algorithm (30 iterations, 5x subsampling)

Aims:

A lot of deconvolution software exists (Huygens, Vaytek, AutoQuant), so it must be carefully checked how much efforts should be done in developing own solutions. It seems perhaps more successful to use commercial software. The challenges lie in the improvement of the accuracy in single protein detection into sub-pixel and sub-resolution range, in a parallel implementation of these methods because of their computational expensiveness, or in an adaptation for path length reconstruction for DIC images.

References:

Agard, D., Hiraoka, Y., Shaw, P., Sedat, J., Fluorescence Microscopy in Three Dimensions, Methods in Cell Biology, 30: 353-377, 1989.

Holmes, T.J., Maximum Likelihood Image Restoration Adapted for Noncoherent Optical Imaging, JOSA-A, 5(5): 666-673, 1988.

Bhattacharyya, S.; Holmes T.J., The ML-Blind deconvolution algorithm: Recent developments, SPIE Vol. 2655

Conchello, J.A. Yu,Q.; Parametric blind deconvolution of fluorescence microscopy, SPIE Vol. 2655

ANALYSIS OF SPOT COUNTING ALGORITHMS IN FLUORESCENCE MICROSCOPY IMAGES

LEILA MURESAN AND BETTINA HEISE

Introduction

A frequent task in medical image processing is to identify spots in fluorescence microscopy images. In the case of micro-arrays, the spots are due to DNA sequences labelled with fluorophores (Cy3 or Cy5), one or more per sequence. For classical microscopes, the high density of fluorophores cannot be well resolved, so only mean intensities of circular regions are computed. With *NanoScout*, single peaks (spots) are counted for each region of interest, and the relative abundance of the sequence (of each specific gene) can be determined even from very small samples.

From the image processing point of view spots can be defined as bright, small, circular features, with little detail at the given resolution. We shall approximate them with a 2D Gaussian profile. The aim of this paper is to compare different approaches of spot detection from the point of view of correctness and complexity. Two straightforward quality measures are spot model variance and image residuum.

1. Spot detection algorithms

Spot detection is only apparently a simple task. If performed manually is tedious and hard to replicate. If performed automatically, it is not straightforward how to choose the parameters in order to obtain a "good" solution. Furthermore, is difficult to define what a "good" solution is. The most popular algorithms in the literature, which will be shortly described and tested below are:

(1) Mathematical morphology
(2) Local thresholding
(3) *À trous* wavelets
(4) Feature based statistics

1.1. **Mathematical morphology.** The tools of mathematical morphology are more appropriate to preprocess images, than to detect spots. By the standard granulometry methods one can determine the spot size and the spot density [1, 2]. If the spot density is too high, the task of counting single spots becomes impossible (classical mean intensity methods have to be applied). We shall refer from now on only to cases when single spot detection makes sense.

1.2. **Local thresholding.** A very simple thresholding algorithm was implemented, which detects a spot at each location (x, y) where the original image intensity reaches a local maximum and the mean intensity in a small window is k times higher than the noise standard deviation.

Date: April 7, 2005.

1.3. *À trous* **wavelets.** The *à trous* wavelet method, described in [4] consists of successive B-spline kernel convolutions. Initially the original image is convolved with the kernel K_0, a B-spline of order 3, $A_1 = Original * K_0$, where :

$$K_0 = \begin{pmatrix} \frac{1}{256} & \frac{1}{64} & \frac{3}{128} & \frac{1}{64} & \frac{1}{256} \\ \frac{1}{64} & \frac{1}{16} & \frac{3}{32} & \frac{1}{16} & \frac{1}{64} \\ \frac{3}{128} & \frac{3}{32} & \frac{9}{64} & \frac{3}{32} & \frac{3}{128} \\ \frac{1}{64} & \frac{1}{16} & \frac{3}{32} & \frac{1}{16} & \frac{1}{64} \\ \frac{1}{256} & \frac{1}{64} & \frac{3}{128} & \frac{1}{64} & \frac{1}{256} \end{pmatrix}$$

The smoothed image is than convolved with a kernel obtained from the kernel of the previous step, by inserting between each line and each column of the old kernel a line and a column of zeros, respectively.

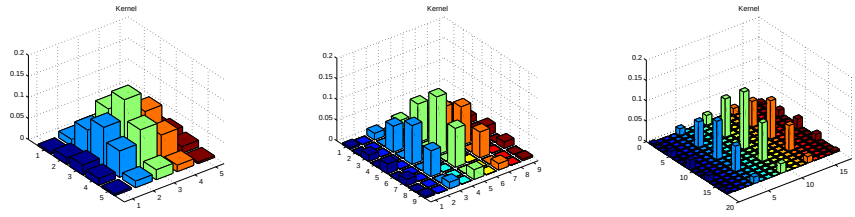

FIGURE 1. Kernels for successive convolutions in the *à trous* method

The wavelet coefficients at step i are: $W_i(x,y) = A_i(x,y) - A_{i-1}(x,y)$. After J steps:

$$Original(x,y) = A_J(x,y) + \sum_{i=1}^{J} W_i(x,y) \tag{1}$$

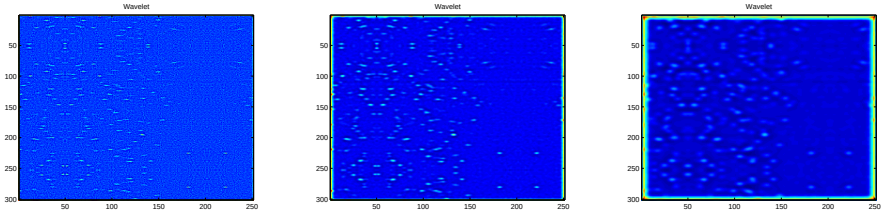

FIGURE 2. Wavelet coefficients for successive scales

The advantage of this image decomposition is the fact that real features tend to be persistent over the scales. So, if we set:

$$W_i(x,y) = 0, \frac{W_i(x,y) - \mu}{\sigma} < 3 \tag{2}$$

and compute

$$Spot_J(x,y) = \prod_{i=1}^{n} W_i(x,y) \tag{3}$$

the bigger the value of $Spot_J(x, y)$ the bigger the likelihood of a spot at location (x, y). Results of the detection can be observed below.

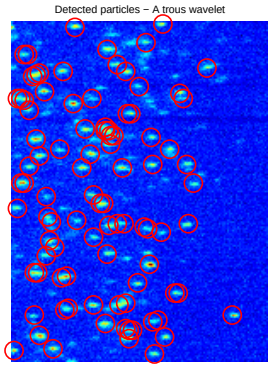

FIGURE 3. Result for *à trous* wavelets method

1.4. **Feature statistics.** The feature based statistics method was thoroughly described in [3]. We only mention that it is based on the modified z-score method for outlier detection. The outliers in several features are forming the set of spot candidates. In order to detect reliably the outliers, robust estimation of mean and variance is used.

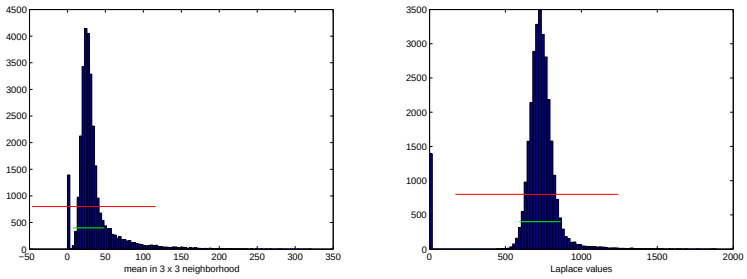

FIGURE 4. Detecting outliers for each feature. With red is represented the standard deviation of the data, with green the robust estimation of the standard deviation around the mean.

This set is clustered in three subsets according to their acceptability level: the class of best, sharpest spots, acceptable spots and the uncertain / out-of-focus spots. The best results were obtained for the Gustafson-Kessel clustering algorithm. The features considered in this paper are: the mean intensity value over a window of size three and the intensity value of the Laplace -filtered image.

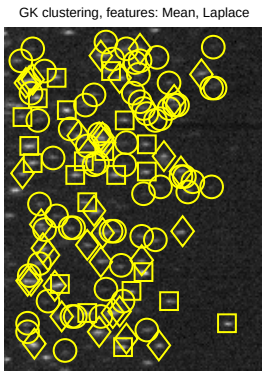

FIGURE 5. Result for the feature statistics method

2. Performance criteria

The performance of spot detector algorithms can be approached from two points of view: correctness and complexity. In this paper we shall concentrate only on the former.

Usually, detection problems in image processing are verified based on test images for which the ground truth is known. The algorithm which minimizes the number of false positive and/or false negatives (or a combination of the two) is preferred. In [6] the criterium to optimize is:

$$AVR = c_1 \cdot \lambda_1 + c_2 \cdot \lambda_2 + c_3 \cdot \lambda_3 + \sum_r c_4(r) \cdot \lambda(r) \tag{4}$$

where λ_1 represents the number of missed spots, λ_2 the number of spuriously detected spots, λ_3 multiple responses, and finally $\lambda(r)$ the localization errors over a distance r.

Unfortunately in most of the medical images ground truth is not known, which leaves the designer of the algorithm with two options: either perform the test on synthetic images or define a reasonable criterium to measure the performance of the algorithm.

A straightforward a approach in the case of real data is to construct a model of the ideal Gaussian profile of spots from the set of detected spots $\boldsymbol{S}$. The simplest way to do this is by pixel-wise averaging the spot intensity values over all elements of $\boldsymbol{S}$, which also holds a denoising effect.

Accuracy can be improved by aligning the spots via a simple fitting algorithm, e.g. intensity weighted fitting as in [5]. The complexity of the fitting algorithm has to be kept low, due to the big number of spots to be analyzed.

In the next step, the image is reconstructed, by placing an ideal spot centered at the coordinates (x_0, y_0) where a spot was detected. If two spots overlap, the respective values are added. The residuum of the image (the pixel-wise squared difference) is a measure of the performance of the algorithm:

$$\sum_{i,j} (Orig(i,j) - Reconstructed(i,j))^2 \tag{5}$$

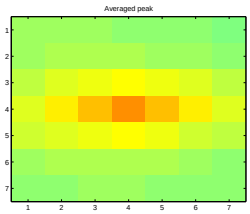

FIGURE 6. Averaged spot model

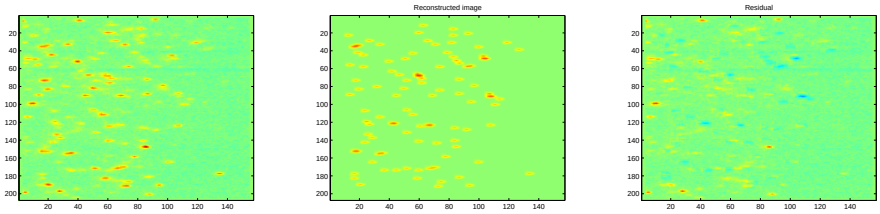

FIGURE 7. Residuum computation

The mean was subtracted from the original image, as a simple background correction. Alternatively, the distribution of volume error per spot can also be informative. We suggest that the mean and the skewness of the error value distribution is related to the validity of the spot model, while the variance characterizes both the quality of the data and the performance of the algorithm.

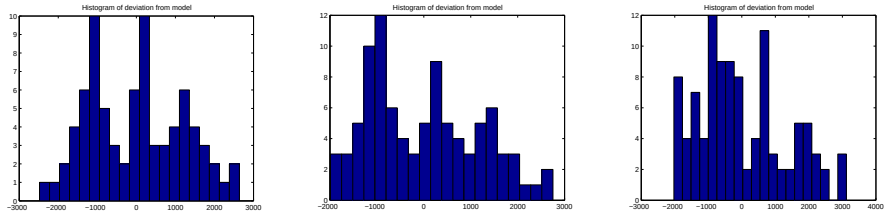

FIGURE 8. Volume error distribution(local thresholding, wavelet and feature statistics method)

A summary of the results is given in table 1. Note the trade-off between the residuum and the spot model (variance) criteria.

Method	Spots	Residuum	Mean	Variance	Skewness
Local threshold	78	2.7429e+007	4.2618e-012	1.4742e+006	0.1920
À trous	93	2.6899e+007	4.7186e-012	1.3694e+006	0.3949
Feature statistics	103	2.0883e+007	1.6115e-013	1.7176e+006	0.4954

TABLE 1. Comparison of the results

3. Conclusions and further work

In this paper we have presented shortly the challenges of spot detection in fluorescence microscopy images, and some algorithms that handle this problem. We have established a way to compare the performance of the algorithm with respect to the correctness of the generated solutions. However further work is needed in order to find a meaningful combination of the spot model and residuum criteria.

It is possible to design iterative algorithms that optimize this correctness criterium, but the amount of data that has to be analyzed (one microarray scan has approximately 2GB) makes the iterative approach unfeasible. However, if the spot model is constructed using only a subimage of the microarray, and this model is valid for the whole image, than an iterative approach might prove useful.

Further refinement of the algorithms are still to be performed (e.g. the implementation of spot alignment in the computation of the Gaussian spot model).

References

1. J. Angulo and J. Serra, *Automatic analysis of dna microarray images using mathematical morphology*, Bioinformatics **19** (2003), no. 5, 553–562.
2. Serra J., *Image analysis and mathematical morphology*, vol. 1, Academic Press, 1988.
3. L. Muresan and B. Heise, *Microarray image analysis*, Tech. report, Dept. of Knowledge-based Mathematical Systems, 2004.
4. J. C. Olivo-Marin, *Extraction of spots in biological images using multiscale products*, Pattern Recognition **35** (2002), 1989–1996.
5. R.E. Thompson, D.R. Larson, and W.W. Webb, *Precise nanometer localization analysis for individual fluortescent probes*, Biophysical Journal **82** (2002), 2775–2783.
6. F. van der Heijden, W. Apperloo, and L.J. Spreeuwers, *Numerical optimisation in spot detector design*, Pattern Recognition Letters **18** (1997), 1091–1097.

Dept. of Knowledge-Based Mathematical Systems, Johannes Kepler University
E-mail address: leila.muresan@jku.at