Towards Evolving Parametric Fuzzy Classifiers Using a Virtual Sample Generation Approach

Holger Hähnel[1] **Arne-Jens Hempel**[1] **Gernot Herbst**[2]

[1]Technische Universität Chemnitz, 09107 Chemnitz, Germany
[2]Siemens AG, Clemens-Winkler-Str. 3, 09116 Chemnitz, Germany

Abstract

Evolving classification models are designed to solve on-line tasks with demands restricting computational power and memory. The present paper proposes an evolving version of an established fuzzy classification approach based on fuzzy pattern classes. The approach incorporates a novel type of virtual sample generation. It creates examples from given parametric model information and thus inverts the classifier's batch learning algorithm. In an evolving environment, virtual examples and real learning data are combined for on-line learning. The main advantage of this approach is that the original learning process retains its applicability while memory demands are reduced significantly. Academic examples demonstrate the feasibility.

Keywords: fuzzy classification, evolving classifier, virtual sample generation

1. Introduction

During the last decade, the necessity for adaptive and evolving models in artificial and computational intelligence applications has been growing [1]. Especially for hardware-based tasks in industrial environments, the model builder is often confronted with the problems of dynamic changes in the system to be modelled. Detecting and integrating these changes on-line into the model while facing restrictions in computational power and memory poses the challenge at hand. Hence, so-called evolving intelligent systems (EIS) came into the focus of recent research activities in many application areas [2]. Notably in classification issues such as in condition monitoring [3, 4], banknote authentication [5, 6], and data streams with drift [7], adaptive and evolving fuzzy approaches have been proven promising.

In the following [2] we distinguish between two concepts which EIS can rely on. On the one hand, the model parameters can be updated incrementally by adding newly available data. In the context of a fuzzy classifier, this may affect e. g. the parameters of the membership functions and results in *adaptive* classifiers. The term *evolving* also includes the second concept, referring to models which are able to update their structure due to new data as well as to changes in the model environment [1]. Structural changes in classification tasks comprise e. g. the reorganisation of fuzzy pattern trees [8], the fusion, split, creation, and disappearance of classes.

In an adaptive context, the original learning data $\boldsymbol{x}_{\text{old}}$ are usually not available, mainly by reason of the memory demands formerly mentioned. We only have model parameters $\boldsymbol{p}_{\text{old}}$ learned from $\boldsymbol{x}_{\text{old}}$. Thus, classical methods perform a mapping $(\boldsymbol{p}_{\text{old}}, \boldsymbol{x}_{\text{new}}) \mapsto \boldsymbol{p}_{\text{new}}$ for updating the parameters on the basis of new data $\boldsymbol{x}_{\text{new}}$. In convenient cases, such mappings contain "exact" recursive expressions, which result in the same parameters that would have been achieved when learning the model from scratch. For example, parameterisations that rely on statistical moments of the learning data suit well for this purpose. For more elaborate parameterisations, e. g. the ones based upon an optimisation scheme, the update mappings mostly end up in heuristics.

In order to avoid heuristics, our approach for an evolving fuzzy classifier goes beyond such conventional methods. Roughly speaking, we invert the classifier's batch learning process. That is to say we generate a set of virtual examples $\boldsymbol{x}_{\text{vs}}$ given only the knowledge of $\boldsymbol{p}_{\text{old}}$ as soon as an update is appropriate.[1] Afterwards, the well-established learning algorithm can advantageously be utilised to compute $\boldsymbol{p}_{\text{new}}$ in terms of $(\boldsymbol{x}_{\text{vs}}, \boldsymbol{x}_{\text{new}}) \mapsto \boldsymbol{p}_{\text{new}}$.

Virtual sample generation (VSG) was originally used to incorporate prior information into a machine learning task [9, 10]. It has been shown that VSG in this context is equivalent to incorporating the prior knowledge as a regulariser. Compared with this, the focus of the paper at hand is on adaptive classifiers, which leads to a different procedure. Besides an incremental model update it also allows the inclusion of evolving phenomena such as the fusion of classes. Moreover, our VSG method might also be beneficial for the integration of prior information. Due to their interpretability, the model parameters can be rather easily determined based on expert specifications. The VSG by means of the mapping $\boldsymbol{p}_{\text{prior}} \mapsto \boldsymbol{x}_{\text{vs,prior}}$ subsequently provides virtual learning data, which serve as prior information similar to [9]. In this article, however, we will focus on incremental learning and adaption of classifiers using chunks of new data.

The remainder of this article is organised as follows: In Sect. 2, the underlying parametric model and learning method are presented, followed by the virtual sample generation approach and its application for an adaptive classifier in Sect. 3. The results are illustrated with the help of toy examples in Sect. 4.

[1]We will not discuss the conditions for some kind of "update necessity" in this paper. Nevertheless, it will become obvious that our method shows its advantages in the presence of occasional update cycles rather than in "any-time up-to-date" tasks [2].

© 2015. The authors - Published by Atlantis Press

2. The Classifier's Foundation: Fuzzy Pattern Classes

We already characterised our type of virtual sample generation as an inversion of the classifier's batch-learning process. Accordingly, we first give a brief description of the basic model for the fuzzy classifier, which we designate as fuzzy pattern class (FPC). Afterwards, the learning process and its parameterisations are presented in order to prepare the subsequent VSG description.

In our opinion, fuzzy classifiers based on FPCs feature a number of advantages in comparison to other approaches, such as Gaussian membership functions. They constitute a compromise between a high data compression, interpretability, and flexibility and have been proven beneficial for many practical applications in the fields transportation science [11], medical diagnosis [12], time series analysis [13], and others.

2.1. Definition and Properties

An FPC is defined on a so-called class space $\tilde{\mathcal{X}}$, not on the original feature space $\mathcal{X} \subset \mathbb{R}^M$. More exactly, a linear mapping t, with

$$\tilde{\boldsymbol{x}} = t(\boldsymbol{x}) = T(\boldsymbol{x} - \boldsymbol{r}), \qquad (1)$$

realises the transformation between $\mathcal{X}$ and $\tilde{\mathcal{X}}$. T denotes a rotation matrix with $T^{-1} = T^\top$, which depends on a set of rotation angles φ_i, $i = 1, \ldots, M-1$ (see Fig. 1). Hereby, one is able to model dependencies between the different features. $\boldsymbol{r}$ is called the FPC's representative and acts as mean w. r. t. the data that describes the class.

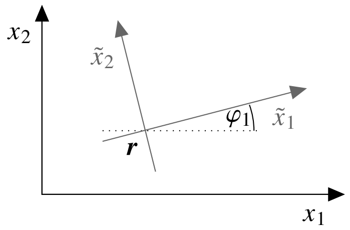


Figure 1: Transformation of the feature space given by (1).

Within the class space, an FPC forms a side-specific multivariate parametric fuzzy set given by the membership function (MF)

$$\tilde{\mu}(\tilde{\boldsymbol{x}}) = \frac{1}{1 + \frac{1}{M}\sum_{j=1}^{M}\left(\frac{1}{b_{j,\,\mathrm{l/r}}} - 1\right) \cdot \left|\frac{\tilde{x}_j}{c_{j,\,\mathrm{l/r}}}\right|^{d_{j,\,\mathrm{l/r}}}} . \qquad (2)$$

It can be understood as an aggregation of M univariate MFs by a compensatory Hamacher operator. In each dimension, an FPC is governed by the left and right class borders $c_{\mathrm{l/r}} \in \mathbb{R}^+$ with their according border memberships $b_{\mathrm{l/r}} \in [0, 1]$ and the fuzziness parameters $d_{\mathrm{l/r}} \in [1, \infty)$. The case $d_{\mathrm{l/r}} \to \infty$ indicates a crisp model. Figure 2 illustrates the univariate case of (2).

Obviously, we can also define the MF on the original feature space $\mathcal{X}$ by setting

$$\mu(\boldsymbol{x}) := \tilde{\mu}(t(\boldsymbol{x})) = \tilde{\mu}(T(\boldsymbol{x} - \boldsymbol{r})) . \qquad (3)$$

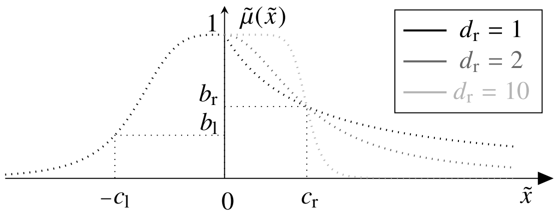


Figure 2: A univariate version of the MF in (2).

When compared with other classification methods such as support vector machines or artificial neural networks, the interpretability of fuzzy approaches has often been stressed. This holds notably for the parameters of an FPC, which are highly comprehensible, e. g. for machine operators and other laypersons.

An FPC can be found on different levels in fuzzy classifiers. First, the modelling of a class by (2) in connection with (1) might be sufficiently accurate. Due to the MF's high flexibility this applies to many problems with convex object morphologies. Beyond that, FPCs can be used as basic building blocks of so-called fuzzy classifier trees in order to model more complex structures, particularly non-convex class shapes [8]. We will restrict the subsequent considerations to the former case.

2.2. Batch Learning of Fuzzy Pattern Classes

First of all, let us look at a general multi-class problem, in which K denotes the number of classes. Within the scope of our concept, we aim to learn membership functions μ_k, for $k = 1, \ldots, K$, from a given set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \subset \mathbb{R}^M \times \{1, \ldots, K\}$ of learning data. If a test datum $\boldsymbol{x} \in \mathbb{R}^M$ belongs to the k^*-th class, the classification shall yield a high truth value $\mu_{k^*}(\boldsymbol{x})$ and low truth values $\mu_k(\boldsymbol{x})$ for $k \neq k^*$. Thus, a way to classify $\boldsymbol{x}$ in a crisp sense would be to declare $k^* = \operatorname{argmax}_k \mu_k(\boldsymbol{x})$. Another possibility is to take the term membership literally by predefining a benchmark $\bar{\mu} \in [0, 1]$ and assigning all classes k with $\mu_k(\boldsymbol{x}) \geq \bar{\mu}$ to the test datum. This results in a non-discriminatory classifier and is especially beneficial for transitional states, e. g. in the field of condition monitoring.

For one-class classification tasks, the benchmark approach seems to be a good choice as well, since there is only one truth value $\mu(\boldsymbol{x})$. This raises the question whether unary and multi-class classification problems can also be treated with the same learning algorithm. In fact, FPCs offer the advantage that each class is learned independently, which splits the above mentioned K-class problem in K one-class tasks. Hence, we restrict the following presentation to one-class classification for data $\mathcal{X}_\mathrm{L} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$. Furthermore, only aspects essential for understanding the VSG are given. A more elaborate description can be found in [14].

Learning of the representative. The representative $\boldsymbol{r}$ of an FPC is given by the mean

$$\boldsymbol{r} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i . \qquad (4)$$

Learning of the rotation matrix. The rotation matrix T of (1) is obtained via a principal component analysis (PCA) of $\mathcal{X}_\mathrm{L}$. By $\tilde{\mathcal{X}}_\mathrm{L} = \{\tilde{\boldsymbol{x}}_i,\ i = 1, \ldots, N,\ \tilde{\boldsymbol{x}}_i = t(\boldsymbol{x}_i)\}$ we denote the data transformed into the class space.

The parameters of the MF in (2) can be learned separately for each dimension of the class space. For the sake of clarity, we therefore omit the dimension index j in the following and consider projections $\tilde{\mathcal{X}}_\mathrm{P} = \{\tilde{x}_1, \ldots, \tilde{x}_N\}$ of $\tilde{\mathcal{X}}_\mathrm{L}$ to any dimension.

Learning of the class borders. The class borders $c_{\mathrm{l/r}}$ are learned by

$$c_\mathrm{l} = -\min_{i=1,\ldots,N} (\tilde{x}_i - c_\mathrm{e}) \quad \text{and} \tag{5}$$

$$c_\mathrm{r} = \max_{i=1,\ldots,N} (\tilde{x}_i + c_\mathrm{e})\ , \tag{6}$$

where $c_\mathrm{e} \geq 0$ is a task specific value (e. g. depending on a sensor resolution or similar) denoted as elementary fuzziness. Alternative parameterisations for $c_{\mathrm{l/r}}$ based on statistical properties of the data are given by [4].

Learning of the fuzziness parameters. The fuzziness parameters can be learned in different ways. While [15] proposes an approach based on the so-called mean distance alteration of adjacent objects, we prefer a statistically motivated method. To this end, we define the sets of left- and right-sided objects

$$\begin{aligned} O_l &= \{\tilde{x}_{\mathrm{l},i}\}_{i=1}^{N_\mathrm{l}} \text{ with } 0 > \tilde{x}_{\mathrm{l},i} \in \tilde{\mathcal{X}}_\mathrm{P},\ i = 1, \ldots, N_\mathrm{l}, \\ O_r &= \{\tilde{x}_{\mathrm{r},i}\}_{i=1}^{N_\mathrm{r}} \text{ with } 0 \leq \tilde{x}_{\mathrm{r},i} \in \tilde{\mathcal{X}}_\mathrm{P},\ i = 1, \ldots, N_\mathrm{r}\,. \end{aligned} \tag{7}$$

The one-sided sample excess kurtoses $\gamma_{\mathrm{l/r}}$ are calculated from the union of the left-/right-sided objects and their mirrored objects w. r. t. zero. Formally, this yields

$$\gamma_{\mathrm{l/r}} = \frac{N_{\mathrm{l/r}} \sum_{i=1}^{N_{\mathrm{l/r}}} \tilde{x}_{\mathrm{l/r},i}^4}{\left(\sum_{i=1}^{N_{\mathrm{l/r}}} \tilde{x}_{\mathrm{l/r},i}^2\right)^2} - 3\,. \tag{8}$$

Afterwards, the excess kurtoses are mapped to $d_{\mathrm{l/r}}$ by

$$d_{\mathrm{l/r}} = 1 + 19^{-\gamma_{\mathrm{l/r}}/1.2}\,. \tag{9}$$

This parameterisation is compatible to the classical version in [15] since $d = 2$ is associated with normally distributed data and $d = 20$ with uniformly distributed data. Moreover, we improve the learning algorithm, mainly with regard to interpretability, without affecting the characteristics of the MF.

With a view to the practicability of the virtual sample generation, the *border memberships* $b_{\mathrm{l/r}}$ are set to 0.5. This constitutes no loss of generality due to the fact that $b_{\mathrm{l/r}}$ and $c_{\mathrm{l/r}}$ depend on each other [14].

3. Virtual Sample Generation for Adaptive FPCs

As already mentioned above, the virtual sample generation (VSG) approach can be seen as an inversion of the batch learning procedure presented in Sect. 2. The main question is: How can we—given a parametric class description—generate a set of virtual examples which, when presented to the batch learning algorithm above, results in a set of parameters similar to the result obtained from learning the original data? One main idea of this paper is to employ a parametric distribution function to generate virtual examples, and map the parameters of the MF to the parameters of the distribution.

In the following, this approach will be presented at first in a general manner (Sect. 3.1), followed by a concrete implementation using a beta distribution for the sample generation (Sect. 3.2). Ideas for the application of the VSG within an adaptive classification workflow are given in Sect. 3.3.

3.1. General Procedure for the VSG

In Sect. 2 it became clear that learning the d parameter ("fuzziness" of the membership function) can be related to the excess kurtosis of the learning data distribution. For the problem of virtual sample generation, we can take advantage of this fact by using a distribution function with a parameterisable kurtosis. When given a certain value of d, this enables us to generate objects of a distribution with a kurtosis similar to the kurtosis of the original learning data. Matching the remaining parameters of the membership function can be obtained by translation, rotation and scaling.

Since the batch learning algorithm from Sect. 2.2 treats all dimensions of the learning data separately as unidimensional problems, the sample generation task will be split into unidimensional problems as well. One aim of this section is therefore to provide a unidimensional deterministic procedure to generate a virtual sample based on a set of parameters of the MF presented in Sect. 2. Learning this generated data set is to result in a MF that matches the original parameters as close as possible.

In general, we consider a family of symmetric cumulative distribution functions (CDFs) F_α, which represent random variables $\tilde{U}_\alpha$ by

$$u = F_\alpha(\tilde{u}) = \mathbb{P}(\tilde{U}_\alpha \leq \tilde{u})\,. \tag{10}$$

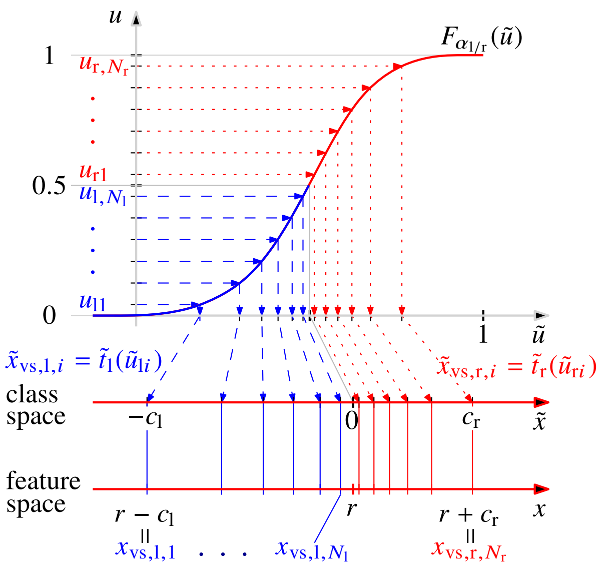
The distribution parameter α shall be related to the excess kurtosis γ by a mapping $\alpha = f(\gamma)$. However, the expectation shall be independent of α, i. e. $\mathbb{E}\tilde{U}_\alpha = m$. We denote the corresponding inverse CDF for fixed α as

$$G(u) = \inf\{\tilde{u} \in \mathbb{R} : F_\alpha(\tilde{u}) \geq u, 0 \leq u \leq 1\}\,. \tag{11}$$

Our method using the inverse CDF G can be considered as a deterministic version of the inverse transformation method, known from the field of pseudo-random number sampling [16]. Followed by scaling and translation, it generates virtual examples in the class space (and therefore in the original feature space). This procedure, which will be divided in several steps in this section, can be visualised as given in Fig. 3. From an abstract perspective, its core can be understood by the scheme

$$u \xrightarrow[\text{step V}]{G} \tilde{u} \xrightarrow[\text{step VI}]{\tilde{t}} \tilde{x}_\mathrm{vs} \xrightarrow[\text{step VII}]{t^{-1}} x_\mathrm{vs}\,. \tag{12}$$

Starting from a deterministic uniform sample with values u, the inverse CDF is used to create values $\tilde{u}$ in a normalised space, which can be translated and scaled to obtain examples $\tilde{x}_{\mathrm{vs}}$ in the class space. Finally, these are retransformed into the original feature space as depicted in Fig. 1, yielding the desired virtual examples x_{vs}.



u
$F_{\alpha_{\mathrm{l/r}}}(\tilde{u})$
1
$u_{\mathrm{r},N_{\mathrm{r}}}$
0.5
u_{r1}
$u_{\mathrm{l},N_{\mathrm{l}}}$
0
u_{l1}
1
$\tilde{u}$
$\tilde{x}_{\mathrm{vs,l},i} = \tilde{t}_{\mathrm{l}}(\tilde{u}_{\mathrm{l}i})$
$\tilde{x}_{\mathrm{vs,r},i} = \tilde{t}_{\mathrm{r}}(\tilde{u}_{\mathrm{r}i})$
class space
$-c_{\mathrm{l}}$
0
c_{r}
$\tilde{x}$
feature space
$r - c_{\mathrm{l}}$
r
$r + c_{\mathrm{r}}$
x
$x_{\mathrm{vs,l,1}}$
$x_{\mathrm{vs,l},N_{\mathrm{l}}}$
$x_{\mathrm{vs,r},N_{\mathrm{r}}}$

Figure 3: Illustration of the virtual sample generation process for the univariate case, starting from a synthetic uniform distribution and finally resulting in virtual examples in the feature space.

For each dimension of the FPC and both the left- and right-hand side of the MF, the algorithm comprises the following steps to generate a virtual sample of size N:

Step I: Recalling the relation between $d_{\mathrm{l/r}}$ and $\gamma_{\mathrm{l/r}}$ in (9), the given parameter values $d_{\mathrm{l/r}}$ can be used to compute the excess kurtoses $\gamma_{\mathrm{l/r}}$, and thus the distribution parameter values $\alpha_{\mathrm{l/r}}$ by means of

$$\alpha_{\mathrm{l/r}} = f\left(\gamma_{\mathrm{l/r}}(d_{\mathrm{l/r}})\right) . \tag{13}$$

Step II: In subsequent steps, we will need the left-sided and right-sided conditional expection of the random variables $\tilde{U}_{\alpha_{\mathrm{l}}}$ and $\tilde{U}_{\alpha_{\mathrm{r}}}$, respectively, given by

$$\begin{aligned} m_{\mathrm{l}} &= \mathbb{E}(\tilde{U}_{\alpha_{\mathrm{l}}} \,|\, \tilde{U}_{\alpha_{\mathrm{l}}} < m) \\ \text{and} \quad m_{\mathrm{r}} &= \mathbb{E}(\tilde{U}_{\alpha_{\mathrm{r}}} \,|\, \tilde{U}_{\alpha_{\mathrm{r}}} \geq m) . \end{aligned} \tag{14}$$

Step III: Due to the membership function's asymmetry, the number of examples on the left- and right-hand side N_{l} and N_{r} will in most cases not be the same and must be determined. Of course it holds

$$N = N_{\mathrm{l}} + N_{\mathrm{r}} . \tag{15}$$

In our approach, the numbers N_{l} and N_{r} are chosen in a way that the resulting virtual examples x_{vs} preserve the location of the class representative r, which means that the examples $\tilde{x}_{\mathrm{vs}}$ in the class space must sum up to zero:

$$0 \overset{!}{=} \sum_{i=1}^{N} \tilde{x}_{\mathrm{vs},\,i} = \sum_{i=1}^{N_{\mathrm{l}}} \tilde{x}_{\mathrm{vs,l},i} + \sum_{i=1}^{N_{\mathrm{r}}} \tilde{x}_{\mathrm{vs,r},i} . \tag{16}$$

However, all virtual examples will be available only at the end of the algorithm. Hence, it is convenient to approximately replace (16) by an equation which relies on the conditional expectations computed in step II. To this end, we make use of the linear transformations $\tilde{t}_{\mathrm{l/r}}$, which map the values $\tilde{u}_{\mathrm{l/r}}$ onto $\tilde{x}_{\mathrm{l/r}}$ (see Fig. 3). It will later become clear that—in anticipation of steps IV to VI—these mappings depend non-linearly on N_{l} and $N_{\mathrm{r}} = N - N_{\mathrm{l}}$, respectively. Thus, the approximative version of (16) is given by

$$0 = N_{\mathrm{l}} \cdot \tilde{t}_{\mathrm{l}}(m_{\mathrm{l}}) + (N - N_{\mathrm{l}}) \cdot \tilde{t}_{\mathrm{r}}(m_{\mathrm{r}}) =: H(N_{\mathrm{l}}), \tag{17}$$

with the non-linear function H. Numerically solving this equation will yield N_{l}, as well as N_{r} via (15).

Step IV: A synthetic uniform distribution of values $\mathcal{U} = \mathcal{U}_{\mathrm{l}} \cup \mathcal{U}_{\mathrm{r}}$ with left side $\mathcal{U}_{\mathrm{l}} = \{u_{\mathrm{l},1}, u_{\mathrm{l},2}, \ldots u_{\mathrm{l},N_{\mathrm{l}}}\}$ and right side $\mathcal{U}_{\mathrm{r}} = \{u_{\mathrm{r},1}, u_{\mathrm{r},2}, \ldots u_{\mathrm{r},N_{\mathrm{r}}}\}$ is set up.

Step V: The distribution information can now be imprinted by applying the inverse CDFs G_{l} and G_{r} from (11) for the respective side. This yields $\tilde{\mathcal{U}} = \tilde{\mathcal{U}}_{\mathrm{l}} \cup \tilde{\mathcal{U}}_{\mathrm{r}}$ with

$$\tilde{u}_{\mathrm{l/r},i} = G_{\mathrm{l/r}}(u_{\mathrm{l/r},i}) .$$

Step VI: The elements of $\tilde{\mathcal{U}}_l$ and $\tilde{\mathcal{U}}_r$ exhibit synthetic distributions of the CDF $F_{\alpha_{\mathrm{l}}}$ and $F_{\alpha_{\mathrm{r}}}$, respectively, on their corresponding domains. Still, they have to be transformed into the class space, i. e. onto an axis of the univariate MF. As pointed out before, this is conducted by affine linear mappings $\tilde{t}_{\mathrm{l/r}}$, which we formalise by

$$\tilde{x}_{\mathrm{vs,l/r}} = \tilde{t}_{\mathrm{l/r}}(\tilde{u}_{\mathrm{l/r}}) = v_{\mathrm{l/r}}\tilde{u}_{\mathrm{l/r}} + w_{\mathrm{l/r}} . \tag{18}$$

Regarding Fig. 3, the offsets $w_{\mathrm{l/r}}$ and the slopes $v_{\mathrm{l/r}}$ result from the conditional equations

$$\begin{aligned} \tilde{t}_{\mathrm{l}}(G_l(u_{\mathrm{l}1})) = -c_{\mathrm{l}}, \quad \tilde{t}_{\mathrm{r}}(G_r(u_{\mathrm{r,N_r}})) = c_{\mathrm{r}} , \\ \text{and} \quad \tilde{t}_{\mathrm{l}}(G_{\mathrm{l}}(0.5)) = \tilde{t}_{\mathrm{r}}(G_{\mathrm{r}}(0.5)) = \tilde{t}_{\mathrm{l/r}}(m) = 0 . \end{aligned} \tag{19}$$

Step VII: For the univariate case, the final step is to transform the virtual examples $\tilde{x}_{\mathrm{vs}} \in \tilde{\mathcal{X}}$ into the feature space $\mathcal{X}$. For the transformation from the class space into the feature space, we can employ the inverse of the tranformation t, defined in Sect. 2.1. We obtain

$$x_{\mathrm{vs,l/r}} = \tilde{x}_{\mathrm{vs,l/r}} + r . \tag{20}$$

In the multivariate case, it is required that all seven steps must be executed for every dimension. In order to cover the entire class space and to prevent correlation between different features of $\tilde{\mathcal{X}}$, we further require a random permutation of the virtual samples in each dimension. Only after the permutation, the multivariate examples $\tilde{\boldsymbol{x}}_{\mathrm{vs}}$ can be transformed (instead of (20)) according to

$$\boldsymbol{x}_{\mathrm{vs}} = T^{\top}\tilde{\boldsymbol{x}}_{\mathrm{vs}} + \boldsymbol{r} . \tag{21}$$

3.2. Concrete Implementation of the Algorithm

When choosing a CDF for the algorithm, it is clear that not all CDFs support all values of the excess kurtosis required for the permissible range of d parameter values ($d \geq 1$, cf. Sect. 2). In this section, we will employ a symmetric beta distribution, which can be parameterised for excess kurtoses $\gamma \in (-2, 0)$. This corresponds to values $2 < d < \infty$, which cover most practical cases. If values $1 < d < 2$ are needed (i. e. $\gamma \in (0, \infty)$), one can choose Student's t-distribution. The special case $d = 2$ ($\gamma = 0$) is covered by the standard normal distribution. For brevity, we will only cover the implementation using the beta distribution in this paper.

The CDF of a symmetric beta distribution with the distribution parameter $\alpha > 0$ is defined as

$$u = F_\alpha(\tilde{u}) = \frac{1}{\mathrm{B}(\alpha,\alpha)} \int_0^{\tilde{u}} (\tau(1-\tau))^{\alpha-1} \mathrm{d}\tau \qquad (22)$$

on its support $[0, 1]$. $\mathrm{B}(\alpha,\beta) = \int_0^1 \tau^{\alpha-1}(1-\tau)^{\beta-1}\mathrm{d}\tau$ denotes the beta function. The excess kurtosis yields

$$\gamma = -\frac{6}{3+2\alpha}\,. \qquad (23)$$

The inverse CDF $G(u)$ as defined in (11) cannot be given in analytic form. Hence, numerical evaluation becomes necessary. The concrete implementation of the algorithm using the beta CDF is presented in the following.

Step I: By means of (9) and (23), the distribution information $d_{\mathrm{l/r}}$ has to be mapped onto the distribution parameters $\alpha_{\mathrm{l/r}}$ via the excess kurtoses $\gamma_{\mathrm{l/r}}$. We obtain

$$\gamma_{\mathrm{l/r}} = -\frac{6}{5}\frac{\ln(d_{\mathrm{l/r}}-1)}{\ln 19} \quad \text{and hence}$$
$$\alpha_{\mathrm{l/r}} = -\frac{3}{\gamma_{\mathrm{l/r}}} - \frac{3}{2} = \frac{5}{2}\frac{\ln 19}{\ln(d_{\mathrm{l/r}}-1)} - \frac{3}{2}\,. \qquad (24)$$

Step II: By virtue of (14) and with $m = \mathbb{E}\tilde{U} = 0.5$ in mind, the left- and right-sided conditional expections can be computed:

$$m_{\mathrm{l}} = \frac{2}{\mathrm{B}(\alpha_{\mathrm{l}},\alpha_{\mathrm{l}})} \int_0^{0.5} \tau^{\alpha_{\mathrm{l}}}\,(1-\tau)^{\alpha_{\mathrm{l}}-1}\,\mathrm{d}\tau \qquad (25)$$
$$m_{\mathrm{r}} = \frac{2}{\mathrm{B}(\alpha_{\mathrm{r}},\alpha_{\mathrm{r}})} \int_{0.5}^{1} \tau^{\alpha_{\mathrm{r}}}\,(1-\tau)^{\alpha_{\mathrm{r}}-1}\,\mathrm{d}\tau \qquad (26)$$
$$m_{\mathrm{l/r}} = \frac{1}{2} \mp \frac{1}{2^{2\alpha_{\mathrm{l/r}}}\alpha_{\mathrm{l/r}}\mathrm{B}(\alpha_{\mathrm{l/r}},\alpha_{\mathrm{l/r}})} \qquad (27)$$

Step III: As denoted, we anticipate steps IV to VI in order compute the left-hand side and right-hand side sample sizes N_{l} and N_{r}. That is to say, we have to specify at least the leftmost value $u_{\mathrm{l}1}$ and the rightmost value $u_{\mathrm{r},N_{\mathrm{r}}}$ of $\mathcal{U}$ for determining the parameters of the linear transformations $\tilde{t}_{\mathrm{l/r}}$ (cf. (19)). It is proposed to choose these values dependent on $N_{\mathrm{l/r}}$ in terms of

$$u_{\mathrm{l}1}(N_{\mathrm{l}}) = \frac{1}{k_{\mathrm{l}}N_{\mathrm{l}}} \quad \text{and}$$
$$u_{\mathrm{r},N_{\mathrm{r}}}(N_{\mathrm{l}}) = 1 - \frac{1}{k_{\mathrm{r}}N_{\mathrm{r}}} = 1 - \frac{1}{k_{\mathrm{r}}(N-N_{\mathrm{l}})}\,, \qquad (28)$$

where $k_{\mathrm{l/r}} > 0$ denote additional parameters, suitable to tune the algorithm. We skip some elementary algebra, which transforms (17) into the concrete non-linear equation for the determination of N_{l}, yielding

$$0 = N_{\mathrm{l}}\left(\frac{2c_{\mathrm{l}}\,[m_{\mathrm{l}} - G_{\mathrm{l}}\,(u_{\mathrm{l}1}(N_{\mathrm{l}}))]}{1-2G_{\mathrm{l}}\,(u_{\mathrm{l}1}(N_{\mathrm{l}}))} - c_{\mathrm{l}}\right) \qquad (29)$$
$$-\,(N-N_{\mathrm{l}})\left(\frac{2c_{\mathrm{r}}\,[m_{\mathrm{r}} - G_{\mathrm{r}}\,(u_{\mathrm{r},N_{\mathrm{r}}}(N_{\mathrm{l}}))]}{2G_{\mathrm{r}}\,(u_{\mathrm{r},N_{\mathrm{r}}}(N_{\mathrm{l}}))-1} + c_{\mathrm{r}}\right).$$

N_{r} can then be computed via $N_{\mathrm{r}} = N - N_{\mathrm{l}}$. Note that there will be a kind of discretisation error, since the optimal number $N_{\mathrm{l/r}}$ will in general not be an integer value.

Step IV: The boundary points of the set $\mathcal{U} = \mathcal{U}_{\mathrm{l}} \cup \mathcal{U}_{\mathrm{r}}$ of equidistantly distributed values $u_{\mathrm{l}i}$ and $u_{\mathrm{r}i}$ have already been given in (28). In a similar manner, but non-parametrically, we propose

$$u_{\mathrm{l},N_{\mathrm{l}}} = \frac{1}{2} - \frac{1}{4N_{\mathrm{l}}} \quad \text{and} \quad u_{\mathrm{r}1} = \frac{1}{2} + \frac{1}{4N_{\mathrm{r}}} \qquad (30)$$

for the "mid points" of $\mathcal{U}$. Thus, each $\mathcal{U}_{\mathrm{l/r}}$ results from a synthetic uniform distribution of $N_{\mathrm{l/r}}$ values with starting point $u_{\mathrm{l/r},1}$ and end point $u_{\mathrm{l/r},N_{\mathrm{l/r}}}$.

Step V: The inverse CDFs for the left- and right-hand side are applied to $\mathcal{U}_{\mathrm{l}}$ and $\mathcal{U}_{\mathrm{r}}$, respectively, using the distribution information computed by (24):

$$\tilde{u}_{\mathrm{l/r},i} = G_{\mathrm{l}/r}(u_{\mathrm{l/r},i}),\quad i = 1,\ldots,N_{\mathrm{l/r}} \qquad (31)$$

An example of the results of steps IV and V can be seen in Fig. 4. The parameters $k_{\mathrm{l/r}}$ have been set to 4 herein.

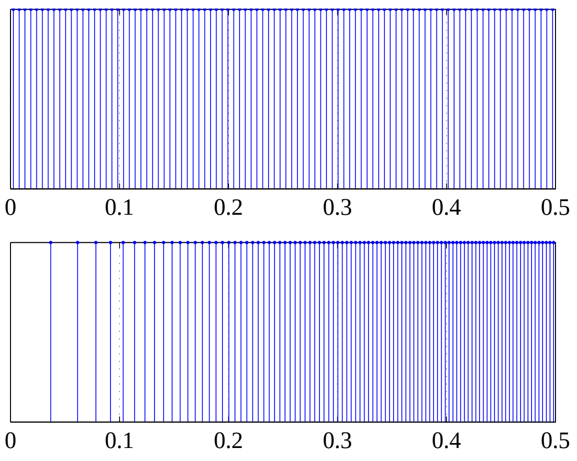


Figure 4: Equidistantly distributed values $u_{\mathrm{l}i}$ of step IV (upper plot) and synthetically beta-distributed values $\tilde{u}_{\mathrm{l}i}$ of step V (lower plot) for $N_{\mathrm{l}} = 94$ and $\alpha_{\mathrm{l}} = 2.19$.

Step VI: The elements of $\tilde{\mathcal{U}}_{\mathrm{l/r}}$ are transformed into the class space, i. e. onto an FPC axis, using the side-specific linear mappings $\tilde{t}_{\mathrm{l/r}}$ from (18). In case of the beta distribution, we achieve

$$\tilde{x}_{\mathrm{vs,l},i} = \frac{2c_{\mathrm{l}}}{1-2\tilde{u}_{\mathrm{l}1}}\,(\tilde{u}_{\mathrm{l}i} - \tilde{u}_{\mathrm{l}1}) - c_{\mathrm{l}} \quad \text{and} \qquad (32)$$
$$\tilde{x}_{\mathrm{vs,r},i} = \frac{2c_{\mathrm{r}}}{2\tilde{u}_{\mathrm{r},N_{\mathrm{r}}}-1}\,(\tilde{u}_{\mathrm{r},i} - \tilde{u}_{\mathrm{r},N_{\mathrm{r}}}) + c_{\mathrm{r}}\,. \qquad (33)$$

Since the execution of *step VII* is independent of the chosen distribution type, we refer to Sect. 3.1 for its description.

3.3. Using VSG for Adaptive Classifiers

As mentioned at the beginning of Sect. 3, we want to construct the virtual sample generation in a way that the set of virtual examples $\boldsymbol{x}_{\mathrm{vs}}$ leads to the same FPC parameters (gained by means of the batch learning procedure in Sect. 2.2) as the original learning data $\boldsymbol{x}$. This marks a crucial point for the usage of VSG in adaptive classifiers since the VSG will be used several times in an on-line learning task and thus errors in the process might accumulate. It turns out that a high accuracy of the VSG in this sense is achieved by tuning the parameters $k_{l/r}$ such that the discretisation error of step III can be nearly compensated.

For the setup of an adaptive classification process based on fuzzy pattern classes and the presented virtual sample generation, the following workflow is proposed:

1. Given an initial set of learning data $\mathcal{X}_{L,0}$, compute the membership function parameters $\boldsymbol{p}_0$ by means of the batch learning procedure of Sect. 2.2. Set the update index $I := 0$.
2. Forget $\mathcal{X}_{L,I}$.
3. When an update is necessary, generate a virtual sample $\mathcal{X}_{vs,I}$ on the basis of $\boldsymbol{p}_I$.
4. Increment $I := I+1$. New learning data for updating has been collected and is denoted as $\mathcal{X}_{new,I}$.
5. Compute the updated MF parameters $\boldsymbol{p}_I$ on the basis of $\mathcal{X}_{L,I} := \mathcal{X}_{vs,I-1} \cup \mathcal{X}_{new,I}$ by batch learning.
6. Go back to 2.

In a straightforward approach, the size of the virtual samples $\mathcal{X}_{vs,I}$ can be chosen equal to the size of $\mathcal{X}_{L,I}$, which means that the class is growing continously without forgetting effects. Moreover, by reducing the sample size of $\mathcal{X}_{vs,I}$, the weight of new learning data can be increased and thus the impact of old data can successively be reduced. Since this practically important strategy is not investigated within the scope of the present paper, it leaves space for future research.

4. Examples

The functional capability of the proposed algorithm shall be illustrated with the help of toy examples. First, the adaption of an FPC according to Sect. 3.3 will be covered with the help of a unidimensional problem. Afterwards, a two-dimensional task is investigated.

4.1. Unidimensional Case

The complete learning data set for the unidimensional example comprises 200 objects which exhibit a deterministic uniform distribution from one to four. Additionally, each object has independently been superimposed with a Gaussian disturbance $\mathcal{N}(0, 0.25)$.

The starting configuration $\mathcal{X}_{L,0}$ consists of the 170 leftmost learning objects and their associated FPC as initial model, see Fig. 5a. Based on this initial model, we add the remaining 30 rightmost learning objects $\mathcal{X}_{new,I}$ in steps $I = 1, 2, 3$, each with ten objects.

The idea behind this approach is to show the adaption of the class borders $c_{l/r}$ as well as the right-hand-sided fuzziness parameter d_r and the class representative r. After each update, we compare the adapted FPC with its batch learning counterpart. Since the reader may not be acquainted with FPCs, we compare the adaption visually, see Fig. 5b and Fig. 5c, and quantitatively, see Table 1. As for the visual comparison, only the initial, first and end configuration will be displayed, due to the space limitations. Besides the batch-learned FPC in blue and the adapted FPC in red, Fig. 5b and Fig. 5c display also the corresponding virtual examples in blue and the new learning objects in red as singletons.

Along with this visual confirmation for the adaption of $c_{l/r}$, d_r, and r, Table 1 displays the values and absolute errors for each update step I in comparison to the corresponding batch learning step.

I	method	r	c_l	c_r	d_l	d_r
1	VSG	2.37	2.16	1.56	5.3	20
	batch	2.36	2.15	1.57	5.4	20
	abs. err.	0.01	0.01	0.01	0.1	0
2	VSG	2.47	2.26	1.81	6.39	16.64
	batch	2.45	2.24	1.83	7.04	18.38
	abs. err.	0.02	0.02	0.02	0.65	1.74
3	VSG	2.58	2.37	2.52	7.54	8.86
	batch	2.56	2.35	2.54	7.94	8.37
	abs. err.	0.02	0.03	0.02	0.4	0.49

Table 1: FPC parameters and adaption errors for the unidimensional example of Fig. 5.

As anticipated, the change in the distribution due to the more noisy objects leads to a smoother form of the class as well as an expansion of the class. Both visual observations in Fig. 5b and Fig. 5c are reflected by the decrease of d_r and the increase of the class borders $c_{l/r}$ (notably c_r), see Table 1.

Errors occur mainly in the d parameter because it is by construction not as robust as r or $c_{l/r}$. However, further experiments showed that there is no systematic error in d such that it averages to zero for continuing updates.

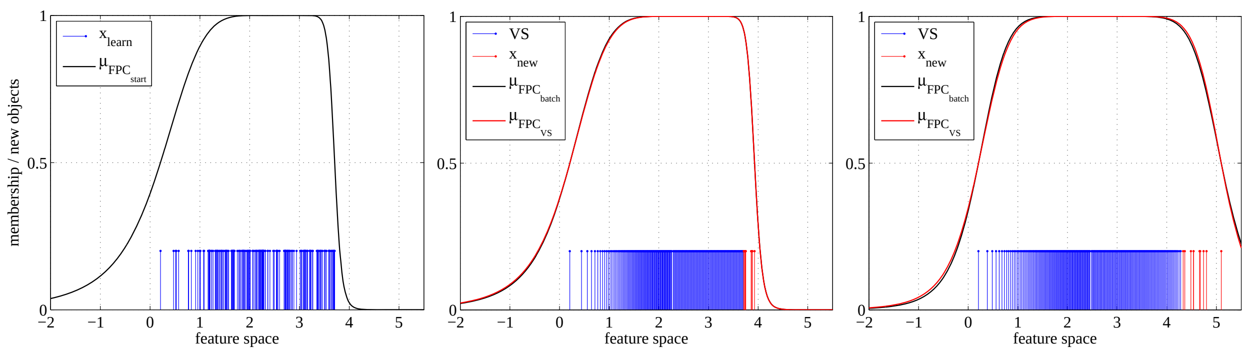
We remark that the elementary fuzziness c_e for the 1D and the following 2D case has been set to zero.

4.2. Two-Dimensional Case

For the two-dimensional example, we first define 300 equidistantly distributed objects on the segment from $(1, 1)$ to $(3, 2)$. Similar to the unidimensional case, these points are disturbed with independent random vectors, yielding the initial learning data $\mathcal{X}_{L,0}$. As disturbance in each dimension, we use a sum of three independent uniform distributions, which has zero mean and variance 1. The according initial FPC model is depicted in Fig. 6a.

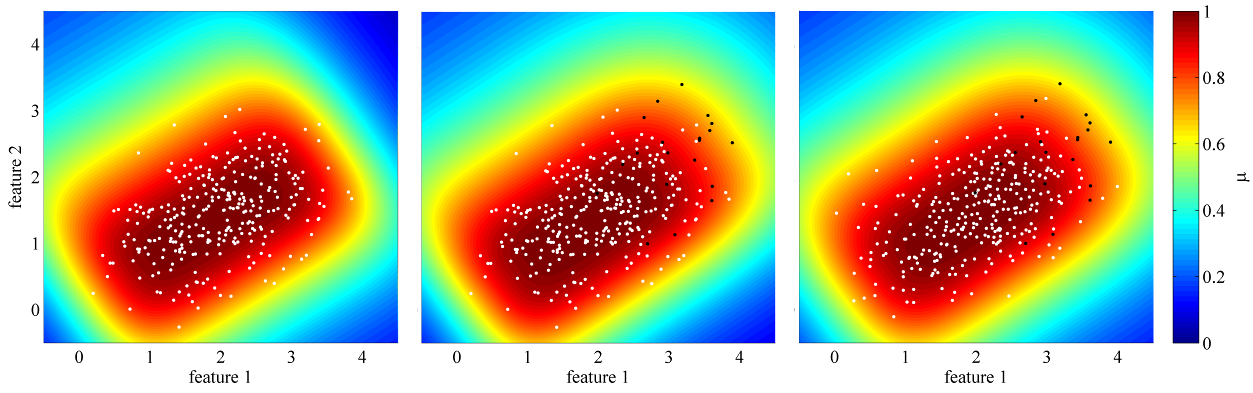
In order to show the classifier's adaption, we only use one step $I = 1$ based on new data $\mathcal{X}_{new,1}$ with sample size 20. The new data are constructed by adding disturbances, in the same way as for the initial learning data, to the point $(3.25, 2.25)$. Figure 6b visualises the sets $\mathcal{X}_{L,0}$ (white) and $\mathcal{X}_{new,1}$ (black) and the model batch-learned from their union. An expansion of the class (i. e. increased $c_{l/r}$) as well as a smoother slope of the MF at the upper right area (i. e. decreased d) can be observed.

The result of the adaption by means of the VSG is displayed in Fig. 6c. The white dots denote the 300 virtual examples $\mathcal{X}_{vs,1}$. High accordance with the batch-learned case can visually be attested.



(a) 170 learning objects and their initial FPC (b) 1st update step with 10 additional objects (c) 3rd update step with 10 additional objects

Figure 5: Example for the adaption of a univariate FPC by means of the VSG in three update steps (2nd not displayed).



(a) 300 learning objects and their initial FPC (b) Batch learning from old and new data (c) Update step with virtual examples and new data

Figure 6: Example for the adaption of a two-dimensional FPC by means of the VSG in one update step.

Table 2 provides the quantitative comparison between the batch-learned and the VSG-based adaptive procedure. Similar to the unidimensional case, the accuracy of the adapted FPC is high w. r. t. the representative $\boldsymbol{r}$ and the class borders $c_{1/2,1/\mathrm{r}}$. Again, deviations occur for the fuzziness parameter d. The remarks passed therefor in the unidimensional case remain valid for the two-dimensional example.

j	method	r_j	$c_{j\mathrm{l}}$	$c_{j\mathrm{r}}$	$d_{j\mathrm{l}}$	$d_{j\mathrm{r}}$
1	VSG	2.05	2.25	2.09	5.62	3.25
	batch	2.05	2.25	2.09	6.59	3.09
	abs. err.	0.01	0.01	0.00	0.97	0.16
2	VSG	1.51	1.31	1.49	2.91	2.36
	batch	1.51	1.32	1.45	3.36	2.27
	abs. err.	0.00	0.01	0.04	0.46	0.08

Table 2: FPC parameters and adaption errors in dimensions $j = 1, 2$ for the example of Fig. 6.

5. Conclusion

We presented an approach for using a novel type of virtual sample generation in the context of an evolving fuzzy classifier. Therefore, we virtually inverted the existing batch learning process for a specific multivariate parametric fuzzy set, called fuzzy pattern class (FPC). The virtual examples have been generated in way that the model information, i. e. the FPC parameters, were retained. The incremental model update for chunks of new data was realisable by means of the familiar batch learning algorithm, but without the need to store the data. This constitutes the main contribution of the article.

Feasibility has been shown in terms of an adaptive learning task for a unidimensional and a two-dimensional toy example. The classifier based on virtual samples was able to adequately reproduce the evolving characteristics of the learning data.

The method is not only restricted to the incremental adaption of classifiers using new learning data. It might also enable us to cope with other evolving phenomena, such as the fusion of similar classes of a classifier. By generating and merging virtual examples for two (or more) similar classes, a resulting class can easily be learned using the existing batch learning algorithm.

Main points for future research include tests with real-world data, an incorporation of forgetting effects, and the investigation of other phenomena of evolving classifiers such as the fusion or the split of classes. A comparison with related adaptive fuzzy and non-fuzzy classifiers might also be fruitful.

References

[1] Moamar Sayed-Mouchaweh and Edwin Lughofer. Prologue. In Moamar Sayed-Mouchaweh and Edwin Lughofer, editors, *Learning in Non-Stationary Environments*, pages 1–17. Springer New York, 2012.

[2] Fernando Gomide and Edwin Lughofer. Recent advances on evolving intelligent systems and applications. *Evolving Systems*, 5(4):217–218, 2014.

[3] Arne-Jens Hempel, Holger Hähnel, and Gernot Herbst. Learning non-convex fuzzy classifiers using single-class SVMs. In *2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2013)*, pages 1–8, 2013.

[4] Holger Hähnel, Arne-Jens Hempel, Uwe Mönks, and Volker Lohweg. Integration of statistical analyses for parameterisation of the fuzzy pattern classification. In F. Hoffmann and E. Hüllermeier, editors, *22. Workshop Computational Intelligence, 6.–7. Dezember 2012*, volume 45, pages 115–131. Karlsruher Institut für Technologie (KIT), 2012.

[5] Walter Dyck, Thomas Türke, Johannes Schaede, and Volker Lohweg. A fuzzy-pattern-classifier-based adaptive learning model for sensor fusion. In *2007 IEEE Workshop on Machine Learning for Signal Processing*, pages 282–287, 2007.

[6] Arne-Jens Hempel, Holger Hähnel, Uwe Mönks, and Volker Lohweg. SVM-integrated fuzzy pattern classification for nonconvex data-inherent structures applied to banknote authentication. In *3. Jahreskolloquium Bildverarbeitung in der Automation (BVAu 2012)*, 2012.

[7] Mahardhika Pratama, SreenathaG. Anavatti, and Edwin Lughofer. An incremental classifier from data streams. In Aristidis Likas, Konstantinos Blekas, and Dimitris Kalles, editors, *Artificial Intelligence: Methods and Applications*, volume 8445 of *Lecture Notes in Computer Science*, pages 15–28. Springer International Publishing, 2014.

[8] Arne-Jens Hempel, Holger Hähnel, and Gernot Herbst. Building hybrid fuzzy classifier trees by additive/subtractive composition of sets. In *15th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2014)*, pages 516–525, 2014.

[9] P. Niyogi, F. Girosi, and T. Poggio. Incorporating prior information in machine learning by creating virtual examples. In *Proceedings of the IEEE*, pages 2196–2209, 1998.

[10] Jing Yang, Xu Yu, Zhi-Qiang Xie, and Jian-Pei Zhang. A novel virtual sample generation method based on gaussian distribution. *Knowledge-Based Systems*, 24(6):740–748, 2011.

[11] Michael Päßler and Steffen F. Bocklisch. Fuzzy time series analysis. In Rainer Hampel, Michael Wagenknecht, and Nasredin Chaker, editors, *Fuzzy Control: Theory and Practice*, pages 331–345. Physica, Heidelberg, 2000.

[12] B. Schmidt, F. Bocklisch, S. M. Päßler, M. Czosnyka, J. Schwarze, J. and J. Klingelhöfer. Fuzzy pattern classification of hemodynamic data can be used to determine noninvasive intracranial pressure. *Acta Neurochirurgica Supplement*, 95:345–349, 2006.

[13] Gernot Herbst and Steffen F. Bocklisch. Recognition of fuzzy time series patterns using evolving classification results. *Evolving Systems*, 1(2):97–110, 2010.

[14] Arne-Jens Hempel and Steffen F. Bocklisch. Fuzzy pattern modelling of data inherent structures based on aggregation of data with heterogeneous fuzziness. In Gregorio Romero Rey and Luisa Martinez Muneta, editors, *Modelling Simulation and Optimization*, chapter 28, pages 637–655. INTECH, 2010.

[15] Steffen F. Bocklisch. *Prozeßanalyse mit unscharfen Verfahren*. Technik, Berlin, 1987.

[16] L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, NY, 1986.