Soft Comput (2007) 11: 193–207
DOI 10.1007/s00500-006-0077-3

ORIGINAL PAPER

Abdelhamid Bouchachia · Roland Mittermeir

Towards incremental fuzzy classifiers

Revised: 8 January 2006 / Accepted: 8 February 2006 / Published online: 26 April 2006

Abstract Fuzzy classification systems (FCS) are traditionally built from observations (data points) in an off-line one shot-experiment. Once the learning phase is exhausted, the classifier is no more capable to learn further knowledge from new observations nor is it able to update itself in the future. This paper investigates the problem of incremental learning in the context of FCS. It shows how, in contrast to off-line or batch learning, incremental learning infers knowledge in the form of fuzzy rules from data that evolves over time. To accommodate incremental learning, appropriate mechanisms are applied in all steps of the FCS construction: (1) *Incremental supervised clustering* to generate granules in a progressive manner, (2) *Systematic and automatic* update of fuzzy partitions, (3) *Incremental feature selection* using an *incremental version* of Fisher's interclass separability criterion. The effect of incrementality on various aspects is demonstrated via a numerical evaluation.

Keywords Incremental fuzzy rule learning · Incremental and supervised clustering · Classification · Incremental feature selection

1 Introduction

Traditionally, in the framework of machine learning, off-line or batch learning aims at extracting some knowledge in the form of a function $\mathcal{F}$ from a set of data points $\mathcal{D}$ characterized by $\mathcal{D} = \{(X_k, y)/y = \mathcal{F}(X_k)\}$, where each data point $X_k = (x_{k1}, \ldots, x_{kn})$ is an n-dimensional vector. Each dimension represents a feature and each component of X_k represents a feature value. The function $\mathcal{F}$ maps each data point in the input space to a corresponding class y in the output space. This is simply a classification formulation which is a traditional formulation of machine learning. The task of classification is therefore given by a function from an input space (a domain) to a set of classes (range). Since we are limited by the number of samples serving to learn $\mathcal{F}$, we can only learn an approximation $\tilde{\mathcal{F}}$ of $\mathcal{F}$.

The task of estimating the function $\tilde{\mathcal{F}}$ (or inducing knowledge) relies on the data set that is provided at the beginning of the learning process. Once this data is exhausted, the process of learning stops. The system can no more learn new knowledge. However, there is a number of applications (i.e. agents, profiling, computer intrusion detection, etc.) where data is not available at once. Data arrives in small batches over long periods of time. The traditional learning scheme cannot deal with such a situation unless the system is retrained with the old and new data. Of course, it is possible to do so whenever new data points arrive to produce an entirely new knowledge from scratch. This approach is known as *revolutionary approach* [26]. But, as the number and the diversity of new data points grow, this non-incremental approach becomes very costly or even intractable. The alternative is to apply an incremental learning scheme which is known as *evolutionary strategy* [26]. This strategy changes some parts of the knowledge to accommodate the new data points. The essence of incremental learning is to allow the learning process to take place over time in a continuous and progressive manner rather than as a one-shot experience [16].

Depending on the type of learning, there are various computational models to induce knowledge. Moreover, this knowledge can be extracted and represented in different forms. Thus, depending on the learning algorithm applied, $\mathcal{F}$ (or $\tilde{\mathcal{F}}$) may take various forms, e.g., weights in neural networks [20], probabilities [14], decision trees [31], and explicit rules [23].

Because rules are easy to understand and to interpret, this work focusses on classification rule extraction from data. FCSs consist of rules of the form if-then with fuzzy (linguistic) antecedents in the if-part and class labels in the consequent part with eventually some confidence factors:

$$\text{If } x_1 \; is \; A_1 \; \wedge \; \ldots \; \wedge \; x_n \; is \; A_n \Rightarrow \tilde{y} = c_j\{\tau_j\} \qquad (1)$$

where x_i, A_i, c_j, τ_j are respectively the features which stand for fuzzy sets, the linguistic values (i.e., domain of the

A. Bouchachia (✉) · R. Mittermeir
Department of Informatics-Systems,
University of Klagenfurt, Klagenfurt, Austria
E-mail: hamid@isys.uni-klu.ac.at

features), the winning class label, and the confidence value associated with the consequent. Various approaches have been applied to generate such rules ranging from pure fuzzy model [10], via hybrid neuro-fuzzy model [27], or hybrid fuzzy-genetic model [11], to rough-neuro-fuzzy model [2].

Traditionally, to build a fuzzy classifier, the input space is partitioned into fuzzy regions (fuzzy relations) which represent the antecedents of rules. To gain more insight into the knowledge disseminated from data, these rules have to be of high quality (accurate, compact, consistent). The quality depends directly on the quality of the fuzzy partitions generated. Fuzzy clustering is a popular technique to achieve partitioning. In general, a projection operation onto the features axes is performed to get the membership functions. This step is crucial for the accuracy of the classifier. The first factor that impacts the quality of rules is foremost clustering. Purity, separability, and compactness of the clusters, which are vital criteria to be observed by the clustering algorithm, depend of course on the nature (i.e., the computational basis) of this algorithm.

Since we are referring to both clustering and classification, it is necessary to draw the difference between these two concepts. Clustering is the process of assigning unlabeled data points to groups using some similarity measure (i.e., distance based, density based, etc.). This process is self supervised. Ideally, two criteria have to be satisfied, namely intra-cluster similarity and inter-cluster dissimilarity. Classification, on the other hand, is the process of assigning labeled data to groups, where labels indicate the classes and are a priori known. If data is continuous (i.e., signal), the process is known as regression. In this work, clustering as a step in the process of FCS construction.

Being concerned in this paper with a particular kind of knowledge discovery from raw data, that is fuzzy rule learning, incremental learning is a key aspect. Rule learning aims at generating rules having the form shown in Eq. 1 which allow to categorize data into distinct classes. Traditional rule learning systems assume that all learning resources are available before the learning phase is conducted. Once this phase is exhausted, the system is no more capable of learning additional rules. In other words, the system is not adaptive in order to accommodate new data and cannot evolve over time.

To overcome this limitation, incremental learning offers mechanisms that help equip rule-based systems with adaptivity. Indeed, incremental learning allows the system to infer rules continually as new data becomes available without forgetting the previously learned ones and without referring at any time to the previously used data. Hence, the system becomes self-adaptive and the acquired knowledge becomes self-corrective. As new data arrives, new rules may be created and existing ones modified allowing the system to evolve over time.

To accommodate this revolutionary concept of incremental learning, we present in this paper an approach that can be seen as a cause-effect process. New data leads to changes in the rule base. These changes are systematically propagated through all layers/steps of the system being used. In a nutshell, the contributions of the paper are threefold:

- Incremental supervised clustering: Given a labeled data set, the first step is to cluster this data with the aim to achieve high purity and separability of clusters emanating from various classes. To achieve that, a clustering algorithm is introduced that is both *incremental* and *supervised*. These two characteristics are vital for the whole process. The resulting cluster prototypes are projected onto each feature axis to generate some fuzzy partitions. Note that traditionally, the fuzzy partitions are produced by clustering algorithms that do not utilize the data labels.
- Fuzzy partitioning and accommodation of change: Fuzzy partitions are generated relying on two steps: Initially, each cluster is mapped onto a triangular partition. In order to optimize the shape of the partitions, the number and the complexity of rules, an aggregation of these triangular partitions is performed. As new data arrives, these partitions are systematically and automatically updated without referring to the previously used data. The consequent of rules are accordingly updated.
- Incremental feature selection: To find the most relevant features, an *incremental version* of Fisher's interclass separability criterion is devised. This mechanism allows to keep an eye on arriving new data and to show how this affects the features that are readily used. As new data arrives, some features may be neglected in the rules and some new features may be included in the rules. The premises of the rules are dynamically and systematically updated. At any time of the life of a classifier, the rule base should reflect the semantical contents of the already used data. To the best of our knowledge, there is no previous work on feature selection algorithms that observe the notion of *incrementality*. All available algorithms work with the assumption that the available data is sufficient to select the most relevant features.

The details of the whole fuzzy modeling process is provided in the following sections. Section 2 overviews the problem of incremental learning in general. Section 3 outlines some related work to fuzzy rule generation. Section 4 introduces the suggested incremental clustering algorithm. Section 5 describes the process of feature selection. The incremental rule induction is presented in Sect. 6. Evaluation of the approach suggested is discussed in Sect. 7. Section 8 concludes and outlines future work.

2 A note on incremental learning

Despite its importance in real life applications, incremental learning (IC) remains a topic at its very earlier stages. Traditionally in machine learning, the learned knowledge is induced from data in a one-shot experiment. This assumes that the whole data, source of knowledge induction, is entirely available during the learning stage. Once this stage

is exhausted, the induced knowledge is no more subject to automatic update. Therefore, the performance of the system depends heavily on the available data used during the learning phase. There are several real-world applications such as intelligent agents, user profile learning, computer intrusion, and software engineering where data becomes available over time. In such applications, it is important to devise learning mechanisms to induce new knowledge without 'catastrophic forgetting' and/or to refine the existing knowledge. The whole problem is then summarized in how to accommodate new data in an incremental way while keeping the system under use. This problem has been studied in the framework of adaptive resonance theory (ART) [8]. This theory has been proposed to deal efficiently with the stability–plasticity dilemma. It is concerned with learning new knowledge without forgetting the previously learned one. Formally, the dilem ma stipulates that a classifier (or any similar machine learning device) is totally stable if it keeps the acquired knowledge in memory without any catastrophic forgetting. However, it is not required or assured that it is able to accommodate new knowledge. On the contrast, a classifier is completely plastic if it is able to continually learn new knowledge without any requirement on preserving the knowledge previously learned. ART classifier networks allow to accommodate new data (plasticity) without forgetting (stability) by generating categories over time whenever the new data is sufficiently dissimilar to the existing categories.

Basically there are two schemes to accommodate new data: either to retrain the classifier from scratch using the old and the new data, known as revolutionary strategy, or to further train the classifier using only the new data which corresponds to evolutionary strategy [26]. The first option fulfills the stability requirement, whereas the second option is a typical IC scheme that is able to fulfill both stability and plasticity. The basic problem in incremental learning is to make a tradeoff between the stability and plasticity ends of the learning memory spectrum as shown in Fig. 1.

As noted in [30], there are many approaches that present some aspects of IC and are used under various names like on-line learning, constructive learning, lifelong learning, concept drift, and evolutionary learning. Therefore, a clear definition of IC turns out to be vital. Here are some of the characteristics that an IC algorithm should satisfy:

- IC should be able to accommodate the plasticity phenomenon by learning knowledge from new data. This data can refer either to already known structure or to a new structure of the data.
- IC should not have access at any time to the previously used data to update the existing classifier. IC can use only the new data.

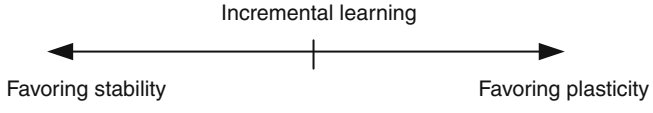

Fig. 1 Incremental learning between stability and plasticity

- IC should be able to observe the stability of the classifier by avoiding forgetting the previously induced knowledge.

Various approaches under the name of "Incremental learning" within the framework of neural networks have been investigated. These range from retraining misclassified sample to various weighing schemes [15,17]. In [30], an incremental learning algorithm that combines an ensemble of multilayer perceptron networks is used to accommodate new data. In [7], an incremental learning algorithm for constructing support vector machine classifiers has been introduced.

On the other hand, incremental symbolic learning has been mainly studied by Michalski and his team [32]. For instance, in [24], a partial memory incremental learning that relies on AQ inductive learning is proposed. It consists of using variable-valued logic as a representation language to generate symbolic rules in the form of term conjunctions. This approach is claimed to be incremental in the sense that the changing learned concepts are incrementally updated. However, some representative examples from the data used in the past are used together with the new samples to update the system. The proposed approach was applied in the context of computer intrusion. In [33] an incremental version of ID3 has been investigated whereas concept drift has been thoroughly investigated in [9].

Many of these algorithms are not truly incremental because at least one of the requirements mentioned earlier is violated. In fact, to avoid the misuse of the term "incremental", Maloof [23] categorizes the learning algorithms in three memory models: no memory, partial memory, and full memory. According to no-memory model, the learned concepts are refined using only the new data which is discarded afterwards. With the partial memory model, after the training phase, some samples are stored for future use (i.e., model not entirely incremental). With the full memory model, all training samples are maintained and used for future use. This latter is obviously not incremental at all.

In this paper, we will apply truly incremental learning to extract fuzzy classification rules. It will be shown how data is discarded once used in an incremental step and how the classifier can incrementally accommodate new rules and rule update without forgetting already learned ones.

3 Fuzzy rule modeling

Fuzzy classification systems consist of fuzzy If–then rules. Each rule describes one region in the n-dimensional feature space and corresponds to one of the H classes to be modeled. The if-part of the rules, known as the rule's antecedent, defines the rule's domain while the rule consequent is the class label from the class label set $\{c_1, \ldots, c_H\}$. The class label can be associated with a confidence factor τ_j. A rule r, R^r, is then given as:

$$\text{If } x_1 \; is \; A_1^r(x_{1k}) \; \wedge \; \cdots \; \wedge \; x_n \; is \; A_n^r(x_{nk}) \text{ then } c^r \qquad (2)$$

where $A_1^r, \ldots, A_n^r$ are the fuzzy sets that stand for antecedent terms. They can be modeled using various membership

functions like: triangular, trapezoidal, Gaussian, etc. In the current work, we will apply two types. Initially triangular membership functions are used and then via an aggregation procedure, trapezoidal membership functions are derived as will be explained later. The degree of activation of a rule r is computed by:

$$\mu^r(x_k) = \prod_{i=1}^{n} A_i^r(x_{ik}). \tag{3}$$

Given a new data point, x_k, entering the system, the output of the fuzzy classifier with respect to this point is the winner class, w, referred in the consequent of the rule with the highest activation degree.

$$\mathcal{L}(x_k) = c^w, \text{ where } w = \underset{r,\, r=1,\mathcal{R}}{\arg\max}\{\mu^r(x_k)\} \tag{4}$$

where $\mathcal{L}$ indicates the class label.

In this paper, we are interested in a more general form of the classification rules that have the following form:

$$\begin{aligned} R^r :\ &\text{If } x_1 \text{ is } A_1^r(x_{1k}) \wedge \ldots \wedge x_n \text{ is } A_n^r(x_{nk}) \\ &\text{Then } c_1^r[b_1^r], \ldots, c_H^r[b_H^r]. \end{aligned} \tag{5}$$

Here, the consequents of the fuzzy rules are associated with some confidence value b_h, called *representativeness degree*. The degree b_h^r associated with each class c_h^r reflects the strength (or the weight) of the class modeled by the rule r. To compute it, a simple proportionality measure is used. Each region (i.e. fuzzy relation) corresponds to a rule, the number of instances in that region for each class is counted and a corresponding degree reflecting the representativeness of each class within that region is computed. Note that by combining rules via certain optimization procedures, we can generate fewer rules whose antecedents terms are connected not only by the AND operator, but also by the OR operator as was done by the author in [6].

To induce rules, first, the incremental clustering algorithm described in Sect. 4 is applied to generate a non-predetermined number of clusters, let it be $\mathcal{C}$. Information about a given class is thus concentrated into $\mathcal{C}$ n-dimensional granules. Because we seek to interpret these granules in terms of linguistic values such as *small*, *medium*, and *large*, the resulting granules are projected onto each dimension (feature) axis so that a feature i, f_i, is patched into linguistically interpretable regions: $f_i = A_1 \times A_2 \times \cdots \times A_{\mathcal{C}}$ equivalent to the number of clusters $\mathcal{C}$. The data representation space will then be organized into n-dimensional linguistically interpretable regions. Each region is actually a fuzzy relation and corresponds to a rule.

We assume in our study that a class can be described by more than one rule. This assumption follows from the fact that the data points of a given class need not necessarily lie contiguously in one compact region of the space. However, if classes are well separated we could represent each class with a single rule. Ideally, the number of rules should be minimal for reasons of computational complexity and foremost for interpretability. To achieve that, a certain number of optimization steps are necessary as discussed in Sect. 7.

On the other hand, the patching has to satisfy some constraints defined in [29], known as *the frame of cognition (FOC)*. This latter stipulates that a fuzzy set (patch) along a dimension must satisfy: normality, typicality, full membership, convexity, and overlap. To fulfill these constraints, we apply two types of membership functions: a triangular, and a trapezoidal membership function for the antecedents A_j.

Trapezoidal forms are generated on top of triangles as a process of optimization. Each triangle, defined by three points (a, b, c), covers a cluster of a given class and is formally described:

$$\mu_{A_{ij}^r}(x_{ik}) = \max\left(\min\left(\frac{x_{ik} - a_{ij}^r}{b_{ij}^r - a_{ij}^r}, \frac{c_{ij}^r - x_{ik}}{c_{ij}^r - b_{ij}^r}\right), 0\right), \tag{6}$$

while a trapezoidal function defined by four points (a, b, c, d) is given as:

$$\mu_{A_{ij}^r}(x_{ik}) = \max\left(\min\left(\frac{x_{ik} - a_{ij}^r}{b_{ij}^r - a_{ij}^r}, 1, \frac{d_{ij}^r - x_{ik}}{d_{ij}^r - c_{ij}^r}\right), 0\right). \tag{7}$$

This process of optimization aims at aggregating clusters that are neighbors and have the same label, meaning that they belong to the same class. This process is necessary because the incremental supervised clustering, that will be explained later, allows to generate more than one cluster within the same class. By aggregating triangles modeling the clusters, the number of regions are reduced and consequently more transparency and compactness of rules is gained. Of course one can perform the aggregation of clusters before generating the triangle partitions. In our experimental evaluation, we did not observe any difference.

Let Fig. 2 plot a synthetic data defined by the characteristics shown in Table 1. The prototypes of the clusters are also displayed in Table 2 and plotted in Fig. 2. Each cluster

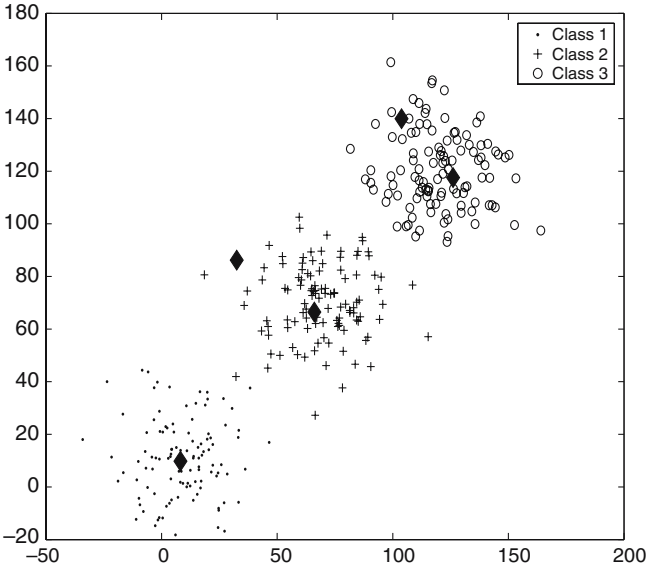

Fig. 2 Synthetic data set

Table 1 Characteristics of the synthetic data set

Chracteristics		μ		Σ	
Features		1	2	1	2
Classes	H_1	10	10	15	15
	H_2	70	70	15	15
	H_2	120	120	15	15

Table 2 Cluster prototypes

Features		1	2
Classes	H_1	8.1425	9.7859
	H_2	66.086	66.456
		32.512	86.2
	H_3	103.81	139.93
		126.1	117.58

Table 3 Triangular membership functions

Features	1			2		
Points	l_{1j}	m_{1j}	r_{1j}	l_{2j}	m_{2j}	r_{2j}
H_1	–34.1	8.1	38.2	–18.2	9.7	37.6
H_2	18.5	32.5	46.5	27.2	66.4	98.2
	32.1	66.0	94.9	80.6	86.2	92.4
H_3	81.6	103.8	117.1	92.4	117.5	138.5
	100.1	126.1	148.6	112.9	139.9	161.4

will be represented by a triangular function (see Table 3 and Fig. 3) and is defined by its center v_j and its left and right endpoints l_j, r_j. Algorithm 1 explains in detail how the triangular membership functions are generated using the clusters.

Algorithm 1 : Generation of triangular functions

for all features **do**
- Project the clusters (l_j, v_j, r_j) onto the current feature axis i. Let this projection correspond to partition j given by A_{ij} which is defined by (l_{ij}, m_{ij}, r_{ij}) corresponding to the left, middle, and right endpoints of the triangular function.
- Sort the projected clusters in an ascending order according to their first endpoint l_j.
for all sorted clusters **do**
- Compute the breakpoints of the triangular function $A_{ij} = (l_{ij}, m_{ij}, r_{ij})$ for the current cluster (l_j, v_j, r_j) as follows:
The projected center v_j corresponds to m_{ij}.
if the current cluster is the first on the feature axis, **then**
$l_{i1} = l_j$
else
if $r_j < l_{j+1}$ // j+1 means next cluster on the axis **then**
$l_{ij} = \frac{r_j + l_{j+1}}{2}$
$l_{i,j+1} = r_j$
else
$r_{ij} = r_j$
$l_{i,j+1} = l_{j+1}$
end if
end if
end for
end for

To optimize this patching, we aggregate the triangular functions modeling clusters that emanate from the same class provided that they are neighbors. The result of this aggregation is shown in Table 4 and plotted in Fig. 4. The aggregation is achieved via a two-step process: fusion based on the second breakpoint (i.e., clusters' centers), and then fusion based on the first breakpoint of triangles as described in Algorithm 2.

Algorithm 2 : Generation of trapezoidal functions

for all features **do**
- Sort the triangles (l_{ij}, m_{ij}, r_{ij}) in an ascending order according to the middle point (i.e., the cluster's center) m_{ij}.
for all sorted triangles **do**
- Compute the breakpoints $(a_{ie}, b_{ie}, c_{ie}, d_{ie})$ of the trapezoidal function from the current triangle (l_{ij}, m_{ij}, r_{ij}) as follows:
$a_{ie} = l_{ij}$
$b_{ie} = m_{ij}$
$c_{ie} = m_{ij}$
$d_{ie} = r_{ij}$
- Get the label of the next triangle, $j = j + 1$, on the axis
while label(current trapeze)=label(next triangle) **do**
$a_{ie} = min(a_{ie}, l_{ij})$
$b_{ie} = min(b_{ie}, m_{ij})$
$c_{ie} = max(c_{ie}, m_{ij})$
$d_{ie} = max(d_{ie}, r_{ij})$
- Get the label of the next triangle, $j = j + 1$, on the axis
end while
end for
- Sort the trapezoidal functions $(a_{ij}, b_{ij}, c_{ij}, d_{ij}$ in an ascending order according to the first breakpoint a_{ij}.
for all sorted trapezoidal functions A_{ij} **do**
- Compute the breakpoints $(p_{ij}, q_{ij}, r_{ij}, s_{ij})$ of the new trapezoidal function l from the current trapeze $(a_{ij}, b_{ij}, c_{ij}, d_{ij})$ as follows:
$p_{il} = a_{ij}$
$q_{il} = b_{ij}$
$r_{il} = c_{ij}$
$s_{il} = d_{ij}$
- Get the label of the next trapeze, $j = j + 1$, on the axis
while label(current trapeze)=label(next trapeze) **do**
$p_{il} = min(p_{il}, a_{ij})$
$q_{il} = min(q_{il}, b_{ij})$
$r_{il} = max(r_{il}, c_{ij})$
$s_{il} = max(s_{il}, d_{ij})$
- Get the label of the next triangle, $j = j + 1$, on the axis
end while
end for
end for

4 Incremental supervised clustering

In order to obtain multi-dimensional granules that serve to partition the feature space with the overall goal of generating fuzzy rules, there are three known methods to do that:

- Partitioning based on interval/grid analysis
- Partitioning by choosing a particular fuzzy membership function, and
- Partitioning based on data specificities such as periodicity as in the case of time-series analysis.

Extensive work is done with respect to the second method. It consists of applying a clustering algorithm that generates some clusters. A partitioning of the feature space is achieved by choosing a membership function and applying it on the obtained clusters. This approach is applied in this work. The key

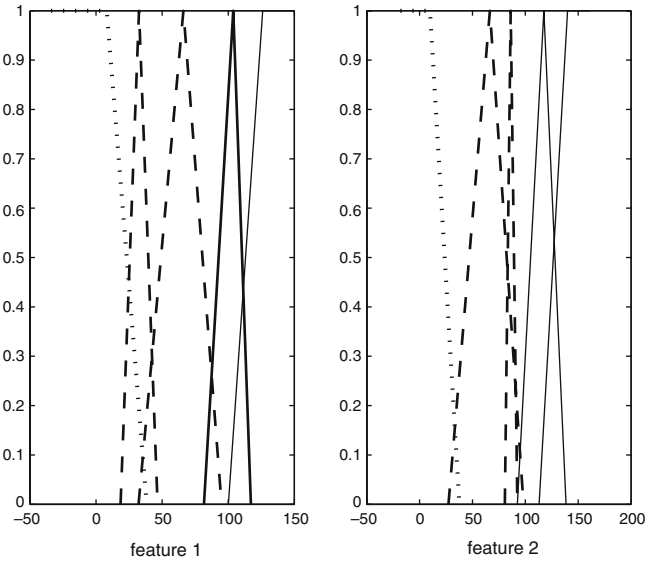

Fig. 3 Derived triangular membership functions

Table 4 Resulting trapezoidal membership functions

Features	Beakpoints	Classes		
		H_1	H_2	H_3
1	p_{1j}	−84.12	18.50	81.62
	q_{1j}	8.14	32.51	103.81
	r_{1j}	8.14	66.08	126.10
	s_{1j}	38.29	94.96	148.61
2	p_{2j}	−68.22	27.23	92.48
	q_{2j}	9.78	66.45	117.58
	r_{2j}	9.78	86.2	139.934
	s_{2j}	37.61	98.29	161.44

point remains the choice of a clustering algorithm to be used. There exists a number of clustering algorithms which are distance-based, volume-based, or distribution-based. Some of the clustering algorithms applied in the context of fuzzy rule generation are fuzzy C-means algorithm [4](e.g. in [10]), Gustafson and Kessel [18], and Gath-Geva [1]. However, these algorithms are fully unsupervised, they do not use any labeling information to generate clusters. In addition, they are batch clustering algorithms and require the pre-specification of the number of clusters to be generated. As we are concerned with incremental learning, we are interested in clustering algorithms that are supervised and incremental at the same time.

In this work, a new algorithm is suggested. It is incremental and the number of clusters is automatically determined during the clustering process. Furthermore, because we are investigating the classification rules, the clustering algorithm has to take labels of data points into account so that clusters are generated within classes. The suggested algorithm consists of steps described in Algorithm 3.

Basically, the algorithm assigns a data point to the closest cluster having the same class label. As a new point is made available, the algorithm computes the distance to clusters of the same class, assigns it to the closest cluster and recomputes the center of this specific cluster. For the sake of illustration, let us reconsider the data set plotted in Fig. 2. Figure 5 shows the effect of σ on the number of clusters generated.

Algorithm 3 : Incremental supervised clustering

1. First data point x_1 is assigned to the first cluster, M_1 whose prototype, v_1, is that data point.
2. Initialize a value σ that indicates the maximal distance between a data point x_k and the prototype v_p of its cluster (it indicates the spread of the cluster). The larger σ, the larger will be the clusters. Therefore, the value of σ has to be reasonable so that the size of clusters gets also reasonable. In the current work, experiments have shown it had not any impact on the final results.
for each next data point x_k **do**
3. Compute the distance $d(x_k, v_p)$ between the data point at hand and each cluster of the same class, hence the supervision aspect of this algorithm.
4. Retain the computed distance d and the index l of the cluster that allows for the smallest distance to that data point.
5. If $d \leq \sigma$, x_k is assigned to cluster l
6. If $d > \sigma$, a new cluster is created.
7. Recompute the prototype of the cluster to which the new point x_k is assigned as follows:

$$v_l^{(t+1)} = \frac{N_l}{N_l + 1} v_l^{(t)} + \frac{1}{N_l + 1} x_k$$

where N_l is the current number of data points belonging to cluster l. The superscript t indicates the time ($v_l^{(t+1)}$ is the center obtained after the arrival of a new data point x_k).
end for

5 Incremental feature selection

To enhance the transparency and the compactness of the rules, it is important that the if-part of the rules does not involve many features. Furthermore, low classification performance

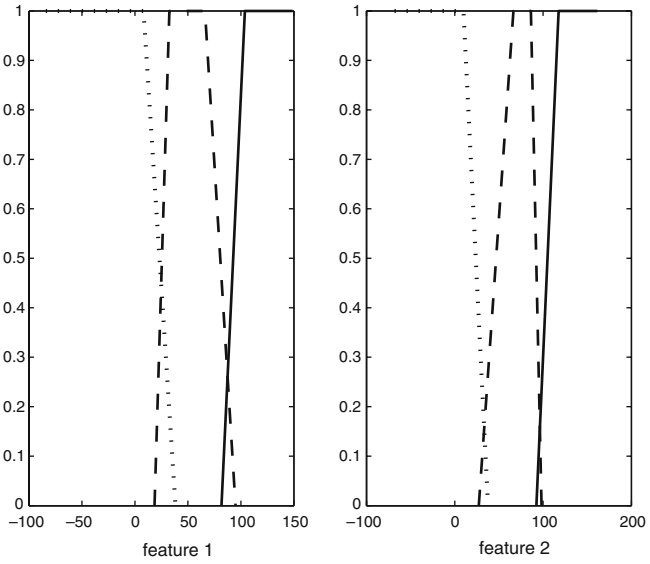

Fig. 4 Derived trapezoidal membership functions

generally results from non-informative features. While there are many ways to achieve this goal like rule reduction [3], rule interpolation [22], hierarchical reasoning [21], the very conventional way to get rid of these features is to apply feature selection methods. Basically, there exit two large classes of feature selection methods [13]:

1. Filters: the idea is to filter out features that have small potential to predict outputs. This technique is done ahead of the classification task. Filters are preprocessing techniques and refer to statistical selection methods such as principal components analysis and single value decomposition.
2. Wrappers: select features that optimize the accuracy of a chosen classifier. Wrappers largely depend on the classifier to judge how well feature subsets are at classifying training samples.

These methods do not lend themselves the possibility to determine the most relevant features from incrementally arriving data. Because the feature selection problem depends foremost on the data being analyzed, these methods all assume that the whole data is available or at least that there is sufficient amount of data for learning the most relevant features. Our literature review indicated that there is no feature selection method that copes with the problem of partially available data.

In this work, we suggest a new approach to deal with feature selection in a dynamic way. This approach assumes that the data used previously will no more be accessible in the future. As a point of departure, we will rely on Fisher's interclass separability criterion [14]. This criterion is based on the between-class scatter and within-class scatter. Given N data points whose mean is v distributed over H classes. Each class j is identified by a center v_j and the number of samples N_j it contains. The between-class scatter is then given as:

$$Q_b = \sum_{j=1}^{H} N_j \left(v_j - v\right)\left(v_j - v\right)^{\mathrm{T}} \tag{8}$$

where the center of the data, v, is given as:

$$v = \frac{1}{N}\sum_{k=1}^{N} x_k$$

while the within-class scatter is given by:

$$Q_w = \sum_{j=1}^{H} Q_j, \tag{9}$$

where Q_j is the covariance matrix which is given by:

$$Q_j = \frac{1}{N_j}\sum_{k=1}^{N_j}\left(x_k - v_j\right)\left(x_k - v_j\right)^{\mathrm{T}}. \tag{10}$$

The total scatter is the sum of between and within scatter matrices:

$$Q_t = Q_b + Q_w. \tag{11}$$

The basic idea of the Fisher interclass separability criterion is to maximize the between-class variance while reducing the within-class variance. The balance between these two scatters is expressed as:

$$J = \frac{\det(Q_b)}{\det(Q_w)} \tag{12}$$

To select features, the criterion (12) is iteratively evaluated on the set of features. At every iteration, the feature that produces the smallest value is removed.

Now, the question that arises is how to equip this selection method with the capability of incrementally selecting features when data arrives over time such that the previously

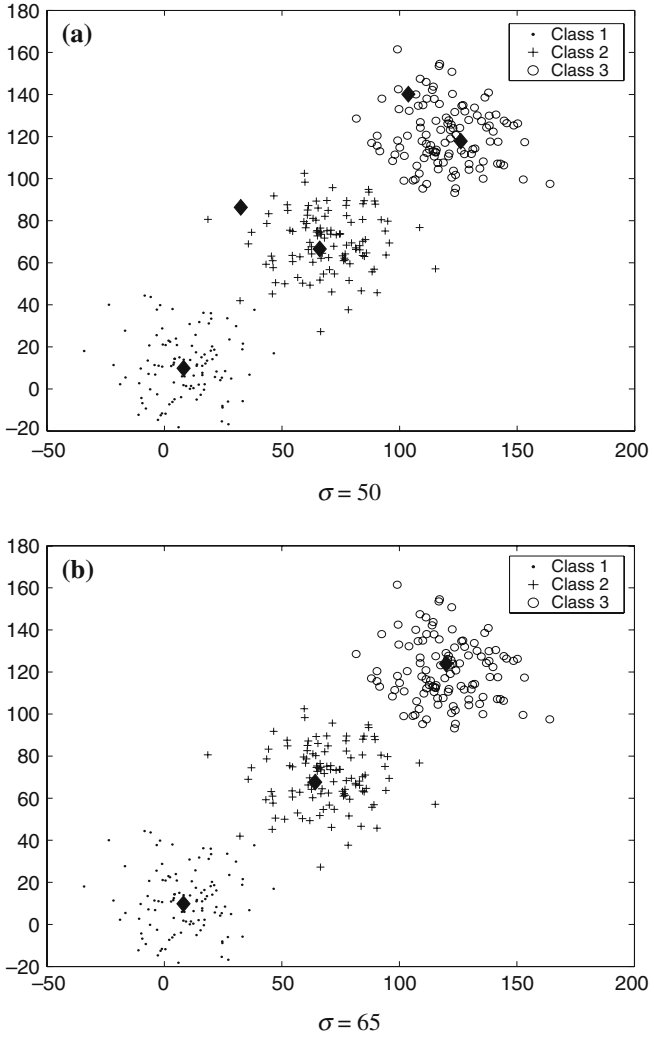

Fig. 5 The resulting number of clusters (prototypes are indicated by filled diamonds) is inversely proportional to the value of σ

used data is no more accessible. To achieve this goal, the within-class and between-class scatter have to be updated incrementally.

Let us assume that at time $(t+1)$, a batch of data points arrives and let a part of it, consisting of n points, affect cluster j. Therefore, the center, the covariance, and the number of points of that cluster change which implies a re-computation of Q_w and Q_b. In the following, we develop the recurrent formulas of the center and covariance matrix that take into account only the new data points, without referring to the previously used data in order to preserve the spirit of incremental learning. Let the cluster j at time (t) consist of N_j data points (for simplicity, the index j in N_j is omitted), $X_j^{(N)} = \{x_1, x_2, \ldots, x_N\}$. Recall that the center of the cluster j, v_j, is defined as:

$$v_j = \frac{1}{N}\sum_{k=1}^{N} x_k. \tag{13}$$

After a new set of points of size n is assigned to the cluster j, i.e., $X_j^{(N+n)} = \{x_1, x_2, \ldots, x_N, x_{N+1}, \ldots, x_{N+n}\}$, the new center is transformed as follows:

$$v_j^{(t+1)} = \frac{1}{N+n}\sum_{k=1}^{N+n} x_k. \tag{14}$$

This can be rewritten as:

$$v_j^{(t+1)} = \frac{N}{N(N+n)}\sum_{k=1}^{N} x_k + \frac{n}{n(N+n)}\sum_{k=N+1}^{N+n} x_k. \tag{15}$$

In other words, the new center can be expressed using the old center as follows:

$$v_j^{(t+1)} = \frac{N}{N+n} v_j^{(t)} + \frac{n}{N+n} w_j, \tag{16}$$

where w_j is the center of the new arriving samples. Then, Eq. 16 can be written:

$$v_j^{(t+1)} = (1-a) v_j^{(t)} + a w_j, \tag{17}$$

where $a = \frac{n}{N+n}$ and $1 - a = \frac{N}{N+n}$.

The recurrent formula (17) indicates that the new center of the cluster affected by adding new data points can be expressed using the previous center of that cluster and the center of the new data points. This means that the data used previously will prevail but not be used for learning new knowledge.

Similarly, the covariance matrix for the cluster j is given by:

$$Q_j^{(t)} = \frac{1}{N}\sum_{k=1}^{N}\left(x_k - v_j^{(t)}\right)\left(x_k - v_j^{(t)}\right)^{\mathrm{T}}. \tag{18}$$

This matrix will be updated as new data points arrive as follows:

$$Q_j^{(t+1)} = \frac{1}{N+n}\sum_{k=1}^{N+n}\left(x_k - v_j^{(t+1)}\right)\left(x_k - v_j^{(t+1)}\right)^{\mathrm{T}} \tag{19}$$

which can be reformulated as follows:

$$Q_j^{(t+1)} = \frac{1}{N+n}\sum_{k=1}^{N}\left(x_k - v_j^{(t+1)}\right)\left(x_k - v_j^{(t+1)}\right)^{\mathrm{T}} + \frac{1}{N+n}\sum_{k=N+1}^{N+n}\left(x_k - v_j^{(t+1)}\right)\left(x_k - v_j^{(t+1)}\right)^{\mathrm{T}} \tag{20}$$

After substituting Eq. 17 in Eq. 20, the updated covariance matrix becomes:

$$Q_j^{(t+1)} = \frac{1}{N+n}\sum_{k=1}^{N}\left(x_k-(1-a)v_j^{(t)}-aw_j\right) \times\left(x_k-(1-a)v_j^{(t)}-aw_j\right)^{\mathrm{T}} + \frac{1}{N+n}\sum_{k=N+1}^{N+n}\left(x_k-(1-a)v_j^{(t)}-aw_j\right) \times\left(x_k-(1-a)v_j^{(t)}-aw_j\right)^{\mathrm{T}} \quad (21)$$

$$Q_j^{(t+1)} = \frac{1}{N+n}\sum_{k=1}^{N}\left(\left(x_k-v_j^{(t)}\right)+a\left(v_j^{(t)}-w_j\right)\right) \times\left(\left(x_k-v_j^{(t)}\right)+a\left(v_j^{(t)}-w_j\right)\right)^{\mathrm{T}} + \frac{1}{N+n}\sum_{k=N+1}^{N+n}\left(\left(x_k-v_j^{(t)}\right)+a\left(v_j^{(t)}-w_j\right)\right) \times\left(\left(x_k-v_j^{(t)}\right)+a\left(v_j^{(t)}-w_j\right)\right)^{\mathrm{T}}. \quad (22)$$

After developing the products within the sum term using $(A+B)(A+B)^{\mathrm{T}} = AA^{\mathrm{T}}+BB^{\mathrm{T}}+AB^{\mathrm{T}}+BA^{\mathrm{T}}$, the expression will be transformed into:

$$\begin{aligned} Q_j^{(t+1)} = \frac{1}{N+n}\Bigg[&\sum_{k=1}^{N}\left(x_k-v_j^{(t)}\right)\left(x_k-v_j^{(t)}\right)^{\mathrm{T}} + \sum_{k=1}^{N}a\left(x_k-v_j^{(t)}\right)\left(v_j^{(t)}-w_j\right)^{\mathrm{T}} \\ &+\sum_{k=1}^{N}a\left(v_j^{(t)}-w_j\right)\left(x_k-v_j^{(t)}\right)^{\mathrm{T}}\Bigg] \\ &+\frac{Na^2}{N+n}\left(v_j^{(t)}-w_j\right)\left(v_j^{(t)}-w_j\right)^{\mathrm{T}} \\ &+\frac{1}{N+n}\Bigg[\sum_{k=N+1}^{N+n}\left(x_k-w_j\right)\left(x_k-w_j\right)^{\mathrm{T}} \\ &+\sum_{k=N+1}^{N+n}b\left(x_k-w_j\right)\left(w_j-v_j^{(t)}\right)^{\mathrm{T}} \\ &+\sum_{k=N+1}^{N+n}b\left(w_j-v_j^{(t)}\right)\left(x_k-w_j\right)^{\mathrm{T}}\Bigg] \\ &+\frac{nb^2}{N+n}\left(w_j-v_j^{(t)}\right)\left(w_j-v_j^{(t)}\right)^{\mathrm{T}} \end{aligned} \quad (23)$$

where $b = a-1$. Taking into account the definition of $v_j^{(t)}$ (Eq. 13), the 2nd, 3rd, 6th, and 7th terms of Eq. 23 will reduce to 0. Consequently, Eq. 23 is transformed into:

$$Q_j^{(t+1)} = \frac{1}{N+n}\left(NQ_j^{(t)}+nZ_j^{(t+1)}+\left(Na^2+nb^2\right) \times\left(v_j^{(t)}-w_j\right)\left(v_j^{(t)}-w_j\right)^{\mathrm{T}}\right) \quad (24)$$

where $Z_j^{(t+1)}$ is the covariance matrix associated with the n new samples. Substituting a and b with their respective expressions in Eq. 24, we get the final expression:

$$Q_j^{(t+1)} = \frac{1}{N+n}\left(NQ_j^{(t)}+nZ_j^{(t+1)}+\frac{Nn}{N+n}\left(v_j^{(t)}-w_j\right) \times\left(v_j^{(t)}-w_j\right)^{\mathrm{T}}\right). \quad (25)$$

Equation 25 indicates that the new covariance matrix of the cluster receiving the new data points can be computed by means of the previous covariance associated with that cluster, the covariance matrix associated with new data, the old center of the cluster and the center of the new data. Again, here the previously used data will not be used to compute the new covariance matrix and consequently to learn new knowledge.

Having shown how to incrementally update the covariance matrix associated with the cluster that is updated, it is easy to apply this straightforwardly in the context of Fisher's interclass separability criterion. The new between-class scatter (after adding new points) requires only to update the number of points of the affected cluster, $N_j^{(t+1)}$, its new center $v_j^{(t+1)}$, and the center of the whole data set using the formula (17) which is equivalent to:

$$v^{(t+1)} = \frac{1}{N+n}\sum_{j=1}^{H}N_j^{(t+1)}v_j^{(t+1)} \quad (26)$$

The between-class scatter is then:

$$Q_b^{(t+1)} = \sum_{j=1}^{H}N_j^{(t+1)}\left(v_j^{(t+1)}-v^{(t+1)}\right)\left(v_j^{(t+1)}-v^{(t+1)}\right)^{\mathrm{T}} \quad (27)$$

The within-class scatter is now given by:

$$Q_w^{(t+1)} = \sum_{j=1}^{H}Q_j^{(t+1)} \quad (28)$$

where $Q_j^{(t+1)}$ is defined by (25).

It is clear that the algorithm can handle batches of any size following the same steps.

To illustrate the effect of adding new data on the selected features, Table 5 shows the process of feature selection using the wine data set that will be presented in Sect. 7. As new data batches arrive the order of importance of the features changes. The incremental feature selection uses the new coming batches to update the order of relevance of the features known up to the moment.

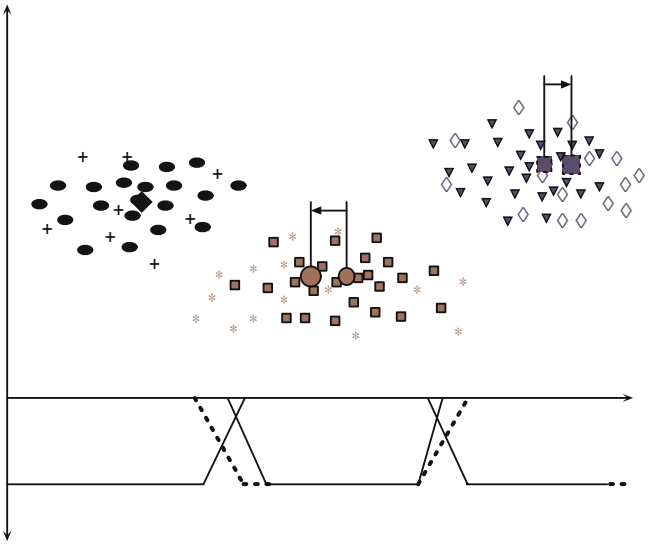

Fig. 6 Updating the membership after the arrival of new data

Table 5 Order of features according to their importance

	Features
Batch 1	5 - 1 - 3 - 4 - 11 - 7 - 8 - 10 - 6 - 9 - 13 - 12 - 2
Batch 2	5 - 1 - 3 - 11 - 4 - 9 - 12 - 2 - 13 - 6 - 10 - 7 - 8
Batch 3	1 - 5 - 9 - 2 - 3 - 8 - 10 - 4 - 12 - 6 - 11 - 7 - 13
Batch 4	8 - 1 - 3 - 9 - 12 - 5 - 10 - 4 - 11 - 7 - 13 - 6 - 2

6 Incremental rule induction

After describing the process of generating rules, the problem of learning new rules or changing the existing ones can be addressed. Incremental learning assumes that the data subject to analysis is not present at once, rather it comes in small batches over time (B_t, $t = 1 \ldots T$). The very important aspect, that is taken into consideration, is that once a data batch has been used to update the rule base, there is no access to it in the future as new ones is made available.

To take the incrementality aspect of data arrival into account, the rule learning algorithm will consist of the following steps:

For each new batch B_t arriving at time t do

1. Assume that all data that came before time t is no more available.
2. Cluster the data batch B_t using the incremental algorithm described in Sect. 4, compute incrementally the new center using Eq. 17, the new covariance matrix according to Eq. 25, and memorize the endpoints of the cluster. Note that only the clusters that have received new data points are subject to further consideration.
3. Based on the new configuration of clusters, the procedure described in Sect. 3 generates the appropriate fuzzy sets relying on an optimization process. If B_t is the first batch to be analyzed, this process identifies which cluster belongs to which trapezoidal partition. A class may consist of many clusters that are aggregated and represented by one trapezoidal membership function.

If B_t is not the first batch, an update operation is triggered. This operation consists of updating the fuzzy partitions corresponding to the clusters receiving new data points. As shown in Fig. 6, the update operation starts by identifying the trapezoidal function to which the modified cluster belongs. There are two steps to be considered:

- The first breakpoint of the trapeze moves to the left if the most left cluster covered by that trapeze receives new points among which there is a point that is smaller than the first breakpoint. In such a case, this left endpoint will be considered as the new first breakpoint of the trapeze. Similarly, the fourth breakpoint will be shifted to a new endpoint if there is a new point that exceeds the current borders of the affected trapeze.
- The second and third breakpoints move to the left or right if the new points push the center of the aggregated clusters covered by that trapeze in a given direction. The amount of shift is simply the difference between the old center and the new center of the aggregated centers. Figure 6 illustrates all cases we can encounter (update only the endpoints of the trapeze, or only the second and third breakpoints, or all of them at the same time).

The two steps described refer to the situation where all new coming data points are assigned to the existing clusters covered by an existing trapezoidal partition. A more complicate situation occurs when the new arriving data points cannot be assigned to the exiting clusters because they are simply far away. As a consequence, two cases can be observed:

- The new generated cluster(s) which belongs to a particular class can be aggregated with some existing clusters from the same class. These clusters are

covered by a trapezoidal partition and therefore the two steps described above can be applied to update that partition.
- If the new cluster, after the projection onto the dimension axis, falls between two clusters emanating from different classes, a new trapezoidal partition is created according to the procedure described in Sect. 3. A consequence to that, the linguistic scale of the rules' antecedents change and a new linguistic term is added. Note that there is no pre-specified linguistic scale. The number of linguistic values depends directly on the number of partitions that corresponds to the number of clusters.

4. For each fuzzy relation corresponding to a rule, compute the representativeness degree of classes as described in Sect. 3. It must be stressed out that the only updated regions are taken into account in this step. These regions are identified after the update of prototypes followed by the update of the partitions.
5. Reformulate the resulting rules in the form of:
   IF x_1 is low and x_2 is low and …and x_n is high THEN $C_1[b_1]$, $C_2[b_1]$, …, $C_m[b_m]$.

This process shows how the update is systematically done and how it is limited to only updated parts of the classifier.

7 Numerical evaluation

To evaluate the proposed approach for incrementally learning classification rules, two benchmark data sets are applied. The wine data and the Wisconsin breast cancer from the UCI Repository of machine Learning Databases [25]. The wine data set describes the chemical composition of 178 samples of the wine. These samples belong to 3 classes of wine and are described by 13 continuous attributes. The Wisconsin breast cancer data consists of 699 samples distributed over two cancer classes: *benign* and *malignant*. There are 16 samples which are incomplete and therefore are not used in our experimental evaluation. This data is described by 9 features.

The data set is divided into two parts: the first part consists of 30% of the data and is meant for testing the classifier; whereas the last 70% is dedicated to training the classifier (i.e. for generating rules). As we are concerned with incremental learning, this latter is divided into four batches of approximately the same size: B_1, B_2, B_3, and B_4 (see Fig.7). The split is done randomly but uniformly so that the same number of samples from different classes is present in all batches.

In the following, we are primarily interested in studying the effect of new data on the rules and related aspects like optimization and incremental feature selection. In addition, the classification performance of the induced classifier is discussed, and a comparison between the incremental classifier against its counterpart that is batch classifier is conducted.

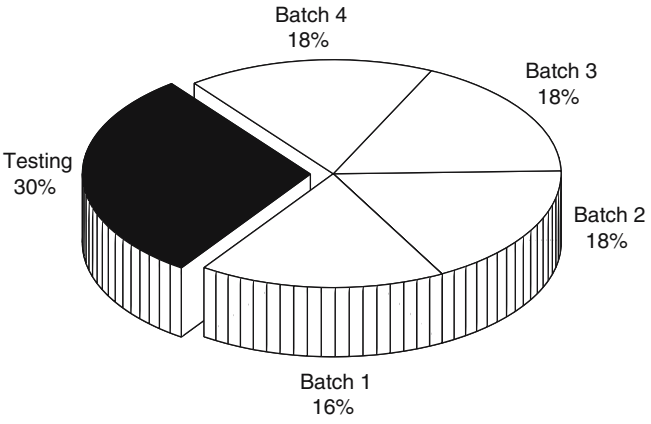

Fig. 7 Split of data

7.1 Rule incrementality

Being the main goal of this study, the incrementality is first discussed from the pure point of view of accommodating new incoming data. Each new batch is introduced to the classifier which will update the existing rules and/or generating new ones during an incremental learning phase. This allow to investigate the sensitivity analysis regarding the introduction of new batches on the rule base. The update may take various forms: generation of completely new rules, generation of partially new rules, and updating the representativeness degrees associated with the classes in the then-part of the rules. Let us first start with the evolution of the number of rules. Table 6 shows the number of rules after the arrival of new batches. As shown, the number of rules generated initially with a small number of data points increases as new batches arrive. The increase pace of rules is more important with the wine data set as with the cancer data although the size of the batches for the former data set is smaller then that of the latter.

The increase in the number of rules is explained by the fact that the regions that initially do not contain points are being populated by new data. However, the resulting number of rules is high due to three facts:

1. No optimization has been performed except the aggregation of triangular partitions of the same class lying in the same neighborhood,
2. All regions have to be taken into account during the whole life of the system because data comes over time and no-instantiated regions can get new points,
3. All features have been utilized; no feature selection has been performed.

The key question is then how would an optimization procedure be triggered given that the obtained rules at a given time are never final due to our fundamental assumption that new data can arrive over time in the future. To cope with this issue, as new data comes, the rule base is updated and an optimized version of it is derived. This latter is used by the classifier until new data arrives.

To generate this optimized version, we will consider a two-stage optimization procedure, namely

Table 6 Effect of the new data batches on rule generation without optimization

Batch no.	Cancer data	Wine data
B_1	44	33
B_2	73	64
B_3	94	97
B_4	117	130

Table 7 Effect of the new data batches on rule generation with optimization

Batch no.	Cancer data	Wine data
B_1	8	31
B_2	8	57
B_3	8	80
B_4	8	101

Table 8 Evolution the order of features as new data arrives

After	Feature order	
	Cancer data	Wine data
Batch 1	4 7 9 1 2 6 8 3 5	5 1 3 4 11 7 8 10 6 9 13 12 2
Batch 2	1 5 8 2 7 9 4 3 6	5 1 3 11 4 9 12 2 13 6 10 7 8
Batch 3	6 4 9 5 3 8 1 2 7	1 5 9 2 3 8 10 4 12 6 11 7 13
Batch 4	9 5 1 8 4 6 7 3 2	8 1 3 9 12 5 10 4 11 7 13 6 2

1. Redundancy elimination: the goal is to reduce fuzzy partitions and rules based on similarity, and
2. Feature selection: the goal is to select features using the method introduced in Sect. 5.

The first optimization procedure consists of four steps: (1) removal of redundant fuzzy partitions, (2) partition combination, (3) removal of redundant rules, and (4) removal of rules with output close to zero. All these steps are based on a similarity measure. Since, we are dealing with trapezoidal membership functions, we will apply the similarity measures described in [12].

- Redundant partitions: they are discovered by computing the similarity of the fuzzy sets describing these partitions to the universe. A fuzzy set A_{ij}^r is removed if:

$$\mathrm{Sim}(A_{ij}^r, U) > \epsilon, \tag{29}$$

  where $\epsilon \in (0, 1)$ and indicates a threshold (a required level of similarity), U indicates the universe that is defined as follows:

$$\forall\, x_k, \quad \mu_U(x_{ki}) = 1.$$

  Any sample has a full membership in the universe of discourse.
- Combination of fuzzy partition: two fuzzy partition are combined if their similarity exceeds a certain threshold:

$$\mathrm{Sim}(A_{ij}^r, A_{ik}^r) > \tau, \tag{30}$$

  where A_{ij}^r,A_{ik}^r are the j^{th} and k^{th} partitions of the feature i in the rule r
- Removal of weakly firing rules: this consists of identifying rules whose output is always close to 0.

$$\mu_i^r < \beta. \tag{31}$$

- Removal of redundant rules: there is redundancy if the similarity between the antecedents of the rules is high exceeding some threshold δ. The similarity of the antecedents of two rules r and p is given as:

$$\mathrm{sim}(A^r, A^p) = \min_{i,\ i=1,n} \{\mathrm{Sim}(A_i^r, A_i^p)\}, \tag{32}$$

  where the antecedents A^r and A^p are given by the set of fuzzy partitions representing the n features $A^r = \langle A_1^r, \ \ldots, A_n^r \rangle$ and $A^r = \langle A_1^r, \ldots, A_n^r \rangle$.

In order to keep a balance between the accuracy and the compactness of the rules, we set the threshold values ϵ, τ, β, and δ to 0.7, 0.7, 0.3, and 0.7, for the cancer data set, while for the wine data set the β parameter is set to 0.18 and the other parameters remain the same. The number of rules resulting after this optimization are shown in Table 7.

It is clear that the first optimization stage dramatically reduces the number of rules for the cancer data (only eight rules), while for the wine data, this number is still high despite the reduction achieved (101 rules after introducing the four batches). As we will see in the next section, this reduction does not affect the classification performance although in general there is tradeoff between transparency achieved by means of this simplification and the accuracy of the classifier.

The second optimization stage relies on the incremental feature selection procedure introduced in this paper. The application of this procedure allows to identify the important features. As shown in Table 8, the degree of feature importance changes as new batches arrive. In fact, some feature highly ranked in the first batch are pushed to the back of the ordered list of features and vice versa (see for instance features 4 and 5). To reduce the number of rules, the tradeoff between the classification accuracy and the transparency of the rules has to be made.

Table 9 shows the number of rules obtained after reducing the number of features for the cancer data set to 7 and 9 features for the wine data set with which we get the best classification performance. It is clearly shown that the incremental feature selection procedure reduces the number of rules. In fact for both data sets, as learning takes place from many batches, the number of rules decreases. Consider the wine data, the number of rules goes from 101 rules obtained via the first stage of the optimization procedure down to 51 after introducing the last batch. For the breast cancer data, the number goes from 8 down to only 4. In summary, the system tends to get the optimal set of rules that cover the whole data, less strong rules are discarded and new rules, if required, are added reflecting the role of incremental learning.

As to the effect of adding new data on the existing rules, the confidence factors associated with the classes that reflect the concentration of samples in the regions modeled by those rules are updated as explained in the previous section. If a rule can handle a part of the incoming data, then its consequent part should be re-weighted with the new data. For the

Table 9 Effect of the new data batches on rule generation with feature selection

Batch no.	Cancer data	Wine data
B_1	5	4
B_2	5	24
B_3	5	25
B_4	4	51

Table 10 Sample of rules discovered from B_1 — cancer data

Rule	x_4	x_7	x_9	x_1	x_2	x_6	x_8	C_1	C_2
1	1	1	1	2	1	1	1	1	0
2	1	1	2	1	1	1	1	1	0
3	1	2	1	2	1	1	1	0.8	0.2
4	2	2	1	2	2	2	2	0.333	0.667
5	2	2	2	2	2	2	2	0	1

Table 11 Sample of rules after introducing B_2 — cancer data

Rule	x_1	x_5	x_8	x_2	x_7	x_9	x_4	C_1	C_2
1	2	1	1	1	1	1	1	1	0
2	1	1	1	1	1	2	1	1	0
3	1	2	2	1	2	1	2	1	0
4	2	2	2	2	2	1	2	0	1
5	2	2	2	2	2	2	2	0	1

sake of illustration, let us consider the cancer data. Table 10 shows the set of rules generated after batch B_1 from the cancer data. Their augmentation after introducing B_2 is illustrated in Table 11. Note that in both tables, the linguistic terms are represented as integers. The intention behind this is to keep the rule as general as possible and to achieve simplicity. When, in the future, these rules involve more than two partitions per input variable, it is easy to designate them by numbers rather than by linguistic values. One can associate a correspondence between these numbers and appropriate linguistic values depending on the feature.

In summary, rules change, over time in an incremental way as new data is made available, with respect to the following aspects:

- Their number,
- Their consequents,
- The number of partitions per input variable, and
- The number of input variables involved in the antecedents of the rules.

7.2 Classification performance

As mentioned in the previous subsection, the number of rules obtained after optimization (see Table 9) after fixing a set of important features. This number was 7 for the cancer data and 9 for the wine data. These features, themselves, are determined on the basis of the accuracy performance of the induced classifier. Table 12 shows how the classification performance of the classifier monotonically increases as it gets new batches of data. This can be explained by the fact that as the amount of data grows the representation space of data is completely covered by the rules generated. It is also worth stressing that the obtained results are very high even though there is no global optimization that involves the whole data, as is traditionally done by means of genetic algorithms and other error minimization techniques. In our case here, this is not possible because the whole data is not available at once and we do not have access to previously used data during the incremental steps.

Table 12 Classification performance of the induced classifier

Batch no.	Cancer data	Wine data
B_1	95.13	85.51
B_2	96.46	93.10
B_3	97.34	96.55
B_4	99.11	96.55

Table 13 Incremental version vs. the batch version: Number of rules

Incremental version			Batch version		
Batch	Cancer	Wine	Available data	Can.	Win.
B_1	5	4	B_1	5	4
B_2	5	24	$B_1 \cup B_2$	6	22
B_3	5	25	$B_1 \cup B_2 \cup B_3$	4	24
B_4	4	51	$B_1 \cup B_2 \cup B_3 \cup B_4$	4	46

7.3 Discussion

In a previous work in the context of incremental rule learning [5,6], a slightly different approach was investigated. It consists of two steps: (1) hierarchical clustering (2) rule induction. In the first step, the incremental supervised Algorithm 3 is applied generating a certain number of clusters. Using FCM, the prototypes of these clustered are then grouped into a predefined number of clusters whose number corresponds to the linguistic scale in which we are interested. In the second step, trapezoidal functions are derived by some partitioning algorithm and rules are then extracted. This approach has been evaluated on Iris plants data set and very encouraging results have been obtained. The approach presented in this paper differs from that approach, where the best classification result obtained on Iris data was 98.7%, in that: no pre-defined linguistic scale is required, a one-step clustering is applied, various aspects of optimization (partition aggregation, rule reduction, feature selection) are considered.

The approach investigated here is highly competitive with non-incremental approaches where full data is present at once as is the case, for instance, in [28] (accuracy is 95.06% for the wine data) and in [19] (accuracy is 99.4% for the cancer data but with 60 rules). Having the whole data available, one can use other optimization techniques like genetic algorithms that might outperform the optimization techniques we use. In the future, one can investigate this aspect thoroughly.

To further appreciate the incrementality classifier presented here, let us compare the classification results and the number of rules obtained earlier (with incrementality,

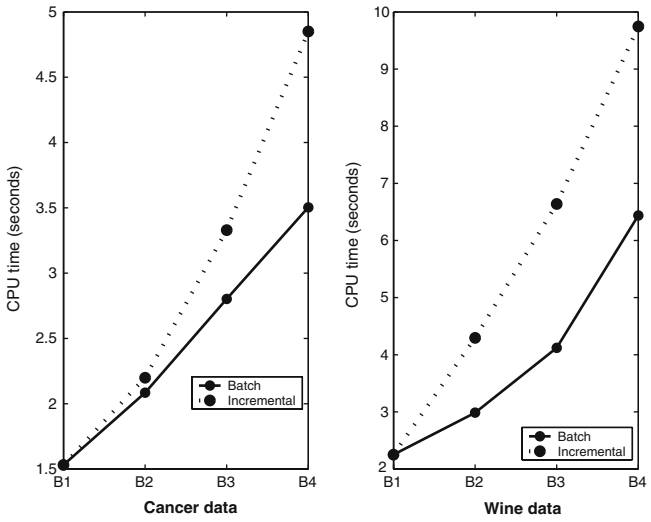

Fig. 8 Execution time of the classifier

see Tables 9, 12) against those obtained with an off-line or batch version of the classifier. To perform this comparison, when an i^{th} batch is used by the incremental version (after learning from the batches $1 \ldots i-1$), the off-line version uses all batches $1 \ldots i$. Tables 13 and 14 show the number of rules and the classification results respectively (For the sake of readability, we reproduce the already known results of the incremental version).

Basically, they show that the results obtained by both versions of the classifier are very similar, though the number of rules with the batch version is somewhat smaller than that obtained with the incremental version. On the contrast, the classification performance resulting from incremental training of the classifier on the cancer data is slightly higher. For the case of the wine data, the batch version is slightly better than the incremental version. The very important fact to be retained here is that the incrementality does not negatively affect the accuracy and the transparency of the classifier.

Another aspect can be discussed here, that related to the time complexity. Although, the training time is not a crucial point given that incremental learning takes place only as new data is made available while the classifier is under use, in the following experiment, we compare the execution time of the incremental and the batch versions. As anticipated, it is clear from the plot (Fig. 8) that the execution time of the batch version is shorter compared with that needed by the incremental version. This is because batch learning is a one-step process, where each task is done once. In incremental learning, the new data is analyzed and the rule base is upgraded via the update operations (described in Sect. 6) that require much time. However, since incremental learning takes place only when new data becomes available, the update does not have any impact on the system. In addition, even when we train the system on new data to obtain a new version of the system, the current version of the system is still operational.

From the perspective of transparency and compactness of the rule base, it is important to note that, using all features allowed to obtain better results with the wine data set achieving up to 98%; while with the cancer data, the reduction of features did not deteriorate the classification results. It is however worth mentioning that in addition to the incremental feature selection based on Fisher's interclass separability method, the redundancy elimination procedure described in Sect. 7 is also of high importance even when solely applied. But, their combination, as is done in this work, has provided better classification results.

Table 14 Incremental version vs. the batch version: Classification accuracy

Incremental version			Batch version		
Batch	Cancer	Wine	Available data	Can.	Win.
B_1	95.13	85.51	B_1	95.13	85.51
B_2	96.46	93.10	$B_1 \cup B_2$	94.41	94.19
B_3	97.34	96.55	$B_1 \cup B_2 \cup B_3$	96.46	96.55
B_4	99.11	96.55	$B_1 \cup B_2 \cup B_3 \cup B_4$	97.90	97.20

A last thing to mention is related to the incrementality aspect, the subject of this paper. The approach investigated here is a "truly" incremental learning approach, where data used previously in an incremental step is never reused in subsequent steps. It considers in an efficient way all conditions an incremental learning algorithm must fulfill (see Sect. 2). Our literature review has shown that most of the algorithms using the wording "incremental" are not really incremental.

8 Conclusion

This paper presents a new approach for learning fuzzy rules in an incremental way. The approach consists mainly of three steps: incremental clustering, rule generation and rule base optimization. The notion of incremental learning is investigated in depth. At all steps of the fuzzy classifier construction,

the incrementality aspect is taken care of. The experiments showed that as new data batches arrive, new rules may be discovered and existing ones may be updated or partially removed. Furthermore, the classification accuracy was found competitive with the no-incremental rule-based classification systems, though the primary goal in this paper is how to equip such systems with plasticity by accommodating new data.

As a future work, it is of interest to investigate some further cases of incrementality. These include situations when arriving batches generate new clusters, or when samples of new batches belong to a different class not known previously. In addition, interpretability issues remain a hot topic and it is important to investigate further reduction mechanisms in addition to those applied in this paper.

References

1. Abonyi J, Szeifert F (2003) Supervised fuzzy clustering for the identification of fuzzy classifiers. Pattern Recognit Lett 24(14):2195–2207
2. Banerjee M, Mitra S, Pal S (1998) Rough fuzy MLP: knowledge encoding and classificationn. IEEE Trans Neural Netw 9:1203–1216
3. Baranyi P, Yam Y (2000) Fuzzy IF-THEN rules in computational intelligence: theory and applications. Fuzzy rule base reduction, Kluwer, Dordrecht, pp 135–160
4. Bezdek J (1981) Pattern recognition with fuzzy objective function algorithms. Plenum Press Englewood cliffs
5. Bouchachia A (2004) Incremental rule learning using incremental clustering. In: Proceedings of the 10th conference on information processing and management of uncertainty in knowledge-based systems, vol 3, Perugia Italy, pp 2085–2092
6. Bouchachia A (2004) Maintaining knowledge bases via incremental rule learning. In: Proceedings of the Inter. Workshop on soft computing for information mining (Located at the 27th German Conference on artifial intelligence), Ulm, Germany pp 51–63
7. Caragea D et al (2001) Architectures for intelligence, chapter Learning in open-ended environments: distributed learning and incremental learning. Springer, Berlin Heiddberg New York
8. Carpenter G, Grossberg S, Rosen D (1991) Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Netw 4(6):759–771, 1991
9. Case J, Jain S, Kaufmann S, Sharma A (2001) Predictive learning models for concept drift. Theor Comput Sci 268(2):323–349
10. Castellano G, Fanelli A, Mencar C (2003) Fuzzy granulation of multi-dimensional data by a crisp double-clustering algorithm. In Proceedings of the 7th multi-conference on systemics, cybernetics and informatics, Orlando, USA
11. Chang X, Lilly J (2004) Evolutionary design of a fuzzy classifier from data. Pattern Recognit Lett 34(4):1894–1906
12. Chen M, Linkens D (2004) Rule-based self-generation and Simplification for Data-driven fuzzy Models. Fuzzy Sets Syst, 142: 243–265
13. De R, Pal N, Pal S (1997) Feature analysis: neural network and fuzzy set theoretic approaches. Pattern Recognit 30(10):1579–1590
14. Duda P, Hart E, Stork D(2001) Pattern classification. Wiley New York
15. Fu L, Hsu H, Principe J (1996) Incremental backpropagation learning networks. IEEE Trans Neural Net 7:757–762
16. Giraud-Carrier C (2000) A note on the utility of incremental learning. AI Commun, 13(4):215–223
17. Grippo L (2000) Convergent on-line algorithms for supervised learning in neural networks. IEEE Trans Neural Netw 11: 1284–1299
18. Gustafson D, Kessel W (1979) Fuzzy Clustering with a Fuzzy Covariance Matrix. In: Proceedings of the IEEE conference on decision and control, pp 761–766
19. Ishibuchi H, Nakashima T, Murata T (1999) Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. IEEE Trans SMC–B, 29:601–618
20. Kasabov N (1996) Foundations of neural networks, fuzzy systems and knowledge engineering. MIT Press New York
21. Kóczy L, Hirota K (1993) Approximate inference in hierarchical structured rule bases. In: Proceedings of the 5th IFSA World Congress (IFSA93), pp 1262–1265
22. Kóczy L, Hirota K (1997) Size reduction by interpolation in fuzzy rule bases. IEEE Trans SMC 27:14–25
23. Maloof M (2003) Incremental rule learning with partial instance memory for changing concepts. In: Proceedings of the international joint conference on neural networks, Los Alamitos, USA pp 2764–2769
24. Maloof M, Michalski R (1995) A method for partial-memory incremental learning and its application to computer intrusion detection. In: Proceedings of the 7th international conference on tools with AI, pp 392–397
25. Merz J, Murphy P (1996) Uci repository of machine learning databases. http://www.ics.uci.edu/- learn/MLRepository.html
26. Michalski (R) (1985) Knowledge repair mechanisms: evolution vs. revolution. In: Proceedings of the 3rd international machine learning workshop, pp 116–119
27. Nauck D, Klawonn F, Kruse R (1997) Foundations of neuro-fuzzy systems. Wiley Chichester
28. Nauck D, Kruse R (1999) Obtaining interpretable fuzzy classification rules from medical data. Artif Intell Medi 16:149–169
29. Pedrycz W, Gomide F (1998) Introduction to fuzzy sets: analysis and design. MIT Press, New York
30. Polikar R, Udpa L, Udpa S, Honavar V (2000) Learn++: an incremental learning algorithm for supervised neural networks. IEEE Trans Systems Man Cybern 31(4):497–508
31. Quinlan J, C4.5 (1993): Programs for machine learning. Morgan Kaufmann San Fransisco
32. Reinke R, Michalski R (1998) Machine intelligence, chapter incremental learning of concept descriptions: A method and experimental results, pp 263–288. Oxford Press, Oxford
33. Utgoff P, ID5 (1988) An incremental ID3. In: Proceedings of the 5th international conference on machine learning, pp 107–120