Knowl Inf Syst (2011) 28:473–489
DOI 10.1007/s10115-010-0324-x

REGULAR PAPER

Fuzzy emerging patterns for classifying hard domains

Milton García-Borroto · José Fco Martínez-Trinidad · Jesús Ariel Carrasco-Ochoa

Received: 2 October 2009 / Revised: 18 April 2010 / Accepted: 28 June 2010 /
Published online: 14 July 2010

Abstract Emerging pattern–based classification is an ongoing branch in Pattern Recognition. However, despite its simplicity and accurate results, this classification includes an a priori discretization step that may degrade the classification accuracy. In this paper, we introduce fuzzy emerging patterns as an extension of emerging patterns to deal with numerical attributes using fuzzy discretization. Based on fuzzy emerging patterns, we propose a new classifier that uses a novel graph organization of patterns. The new classifier outperforms some popular and state of the art classifiers on several UCI repository databases. In a pairwise comparison, it significantly beats every other single classifier.

Keywords Fuzzy emerging patterns · Emerging patterns · Supervised classification

1 Introduction

A pattern is an expression in some language that describes a collection of objects. In Pattern Recognition, we frequently detect and extract patterns from data to achieve certain goals. Bongard [1] was one of the pioneers using explicit patterns in supervised classification. He introduced the concept of complex features for 2-class problems. Complex features are three-attribute conjunctive properties, which appear frequently in a class and do not appear in the other. A complex feature has the following form:

$$(Attribute_1 = value_1 \wedge Attribute_2 = value_2 \wedge Attribute_3 = value_3)$$

Bongard also introduces KORA-3, a new kind of classifier based on complex features. To classify an unseen object, KORA-3 combines the weights of the complex features that the

M. García-Borroto (✉)
Centro de Bioplantas, Carretera a Moron km 9, Ciego de Avila, Cuba
e-mail: milton.garcia@gmail.com

J. F. Martínez-Trinidad · J. A. Carrasco-Ochoa
Instituto Nacional de Astrofísica, Óptica y Electrónica,
Luis Enrique Erro No. 1, Sta. María Tonanzintla, C.P. 72840, Puebla, México

query object contains. Finally, the class with the highest support is assigned to the object. To extend the applicability in real problems, some authors [2] extend the concept of complex features for multi-class problems as multiple-attribute conjunctive properties appearing "enough" in a class, and not too much in the other classes.

Michalski and Step [3] introduce the use of explicit patterns in unsupervised classification (clustering). They find a kind of pattern named l-complex to describe the elements in each cluster, using conjunctions of attribute value properties named selectors $[Attribute \# Set]$, where Set is a subset of values in the domain of $Attribute$, and # is any of a predefined set of operators $>, <, \geq, \leq, =, \neq, \in, \notin$.

Dong and Li [4] boost research about pattern-based classifiers with the development of emerging patterns (EPs). An emerging pattern is a combination of attribute values that occurs mostly in a class, which barely appears in the remaining classes; so the presence of a pattern in a query object gives some evidence about the class the object should belong to. After Dong and Li's work, many authors propose algorithms to extract and use emerging patterns for supervised classification [5].

There are two main families of emerging pattern–based classifiers, according to when the pattern discovery takes place. The first family, introduced by Li et al. [6], makes a lazy discovery. Thus, in the classification stage, the classifier searches for patterns contained in the query object. In contrast, the second family, introduced by Dong et al. [7], searches all emerging patterns in the training stage, and later uses them to classify.

Extracting emerging patterns from a training sample is still a challenge because of three main reasons:

1. The anti-monotonic property [8] used for frequent itemset discovery does not hold for emerging patterns, so common algorithms like Apriori cannot be used.
2. There are many potential candidates in high dimensional databases.
3. Frequently, continuous attributes cannot be rationally compared using the equality operator. For example, the values 3, 2.999 and 3.001 are not equal, but they might belong to the same pattern.

Lazy emerging pattern classifiers compare numerical attributes using a neighborhood-based criterion. They use an a priori selected neighborhood factor α, thus two values x_1, x_2 are considered as equal if $x_1 \in [x_2 - \alpha, x_2 + \alpha]$. Nevertheless, extracting all the emerging patterns from a database in the training stage cannot use such a simple solution. On the other hand, most of the methods for mining emerging patterns in the training stage apply *data discretization* on all numerical attributes. Data discretization is the process of transforming numerical attributes into a finite set of intervals, causing minimal loss of information [9]. However, discretizing individual attributes to extract emerging patterns could have the following serious drawbacks:

- Emerging patterns are combinations of values that must appear simultaneously, thus a priori discretizing individual attributes could deteriorate the quality of the patterns. An example of this behavior appears in Table 1, where for *tae* and *wpbc* databases, the SJEP classifier [10] transforms most numerical attributes into a single-valued non-numerical attribute. This way, those attributes are virtually discarded because they cannot appear in any pattern, so the SJEP classifier was unable to classify any object.
- Discretization usually defines crisp boundaries, therefore the object (3, 5) might match a pattern and (3.001, 5) might not.

Fuzzy Set logic [11] is a solution to the discretization problems stated above. Thus, fuzzy sets have contributed to the improvement of many mining tasks, like mining fuzzy association rules from uncertain data [12].

Table 1 SJEP accuracy versus 3-Nearest Neighbor accuracy in some UCI databases

Database name	SJEP accuracy	3-Nearest neighbor accuracy
Autos	12.31	68.24
Glass	20.43	69.03
Iris	75.33	96.59
Labor	55.33	90.67
Lymph	43.86	85.90
Tae	0	68.75
wpbc	0	72.30

In this paper, we introduce a new kind of emerging pattern, named fuzzy emerging pattern (FEP). Fuzzy emerging patterns are patterns formed by fuzzy selectors $[Attribute \in FuzzySet]$, joined by a fuzzy AND operator. This way an object satisfies a given pattern to a certain degree according to the degree the object attribute values satisfy the property expressed in the pattern.

To efficiently extract fuzzy emerging patterns from a database, we will use a set of fuzzy decision trees induced with a new algorithm that includes the use of linguistic hedges. These hedges allow modifying the semantic of the initial fuzzy discretization to satisfy the semantic of the different labels in the training sample [13]. Finally, we propose a new classifier based on fuzzy emerging patterns, which includes a novel mechanism for aggregation of single pattern votes. This classifier outperforms many state of the art classifiers in most of the tested databases.

The rest of the paper is organized as follows. In the next section, we present a brief review of previous work about emerging pattern classifiers and fuzzy decision tree induction. In Sect. 3, we introduce the concept of fuzzy emerging patterns, the algorithm used to find them, and the classifier. In Sects. 4 and 5, we show the results of the experiments and our conclusions, respectively.

2 Previous work

2.1 Emerging patterns

In 1999, Dong and Li [4] introduce the concept of emerging pattern for a two class problem, as an itemset with significantly more support in one class than in the other class. The first EP-based classifier, named CAEP [7], appears shortly. CAEP discretizes each numerical attribute in five bins using the equal bin population method. The classifier adds the support of each pattern supported by a query object and normalizes this value using the mean vote per class in the training sample. This is known as *aggregate score*.

In 2000, Li et al. [6] introduce the first lazy emerging pattern classifier. They only use general patterns, which are minimal according subset inclusion, to reduce the amount of comparisons and boost classifier speed. Those patterns are known as essential Jumping Emerging Patterns [14]. To classify a query object, the algorithm aggregates the support of the matching patterns using the *compact summation* method. The *compact summation* in class C_i is the percentage of instances in C_i containing one or more of the patterns. Classifiers in this family use a neighborhood-based criterion to compare two numerical values, thus they do not use discretization.

In 2006, Fan and Ramamohanarao [10] introduce SJEP, which includes some modifications on the mining process and uses a different discretization procedure based on the Entropy method [15]. Appice [16], on the other hand, discretizes numerical attribute values through equal width intervals, using a fixed number of bins.

No matter the diversity in the proposed solutions to mine emerging patterns with numerical attributes, no global solution exists. Every a priori discretization scheme works bad in some databases. For example, classifying some databases in the UCI Repository of Machine Learning [17], SJEP [10] using the Entropy [15] method, obtains the results shown in Table 1.

These poor results, compared to the simple 3-Nearest Neighbor [18] classifier, are due to a wrong discretization of numerical attributes. In those databases, the Entropy discretization method transforms most numerical attributes into a single-valued non-numerical attribute, which cannot appear in any pattern.

2.2 Fuzzy decision trees

Inducing fuzzy decision trees (FDT) [19] has been an active research area for many years. Some FDT are simple extensions of hard domain classifiers to fuzzy domains, like Fuzzy ID3 [20] and its variations [21]. On the other hand, some authors have created novel schemes of decision tree induction specially designed for the fuzzy case. An example is LR-COG [22], which uses an induction procedure based on the trapezoid center of gravity, using an information entropy minimization heuristic.

Linguistic hedges ("very", "often", "somewhat") are commonly used in learning algorithms to fix the discretization of continuous attributes. They can be very useful because, generally speaking, a wrong initial discretization of some attributes could become a strong limitation in the classification quality [13].

Following previous ideas, we introduce the definition of fuzzy Emerging Patterns and a novel extraction procedure, based on a new fuzzy decision tree induction algorithm. Additionally, we introduce a new classifier based on fuzzy patterns and a novel vote-aggregation scheme.

3 Classifying with fuzzy emerging patterns

First, we introduce the concept of *Fuzzy Pattern*.

Definition 1 A *Fuzzy Pattern* is a conjunction (using a T-Norm) of *selectors* $[Attribute \in FuzzySet]$, where $\in$ is the membership of the $Attribute$ value to $FuzzySet$; in this paper, these selectors will be named *F-selectors*.

For example, $[Temperature \in hot] \wedge [Humidity \in normal]$ and $[Outlook \in sunny] \wedge [Windy \in true]$ are fuzzy patterns. Additionally, any number of linguistic hedges can modify each $FuzzySet$, like in $[Temperature \in very(hot)] \wedge [Humidity \in somewhat(normal)]$.

Instead of the simple "an object supports a pattern" with the classical emerging patterns, in the case of fuzzy patterns, every object supports every fuzzy pattern in certain *degree*, which can be eventually zero. The individual fuzzy support of a fuzzy pattern fp with respect to an object o, denoted as $fsup(o, fp)$, can be calculated as the minimum value (or another T-norm) of the membership (μ) of every attribute value of o to the fuzzy set in the respective F-selector (fs) of the fuzzy pattern fp:

$$fsup(o, fp) = \min_{fs \in fp} \{\mu_{fs}(o)\}$$

In a problem with multiple classes, each fuzzy pattern fp has a different support in every class C_i, which is calculated as the sum of the individual fuzzy support for all objects in C_i.

$$FSup(fp, C_i) = \sum_{o \in C_i} fsup(o, fp)$$

In order to measure the relevance of each fuzzy pattern for classification, we introduce the concept of *Trust*.

Definition 2 The *trust* of a fuzzy pattern fp is

$$Trust(fp) = \frac{\max_{C_i} FSup(fp, C_i)}{\sum_{C_i} FSup(fp, C_i)}$$

The $Trust$ of a fuzzy pattern measures the ratio of the support of fp in the class with the highest support, with respect to the total support of fp in all the classes. We can understand the $Trust$ as the degree of membership of a given pattern to the fuzzy set of "good patterns for classification". It is a measure that evaluates the power a pattern has to discriminate between classes, and we will use it to select fuzzy emerging patterns. A $Trust$ above 0.5 means the pattern has higher support in the highest supported class than in the combined remaining classes. This is why we use this value as cutting-point to determine which patterns are good for supervised classification. It is worth to mention that a pattern P with support in a class significantly higher than the support in the remaining classes has a higher $Trust$ than a pattern Q with lower differences; since the $Trust$ is used as voting weight, P has a higher influence in the final classification of the objects than Q.

Definition 3 A *Fuzzy Emerging Pattern* (FEP) is a fuzzy pattern with $Trust > 0.5$

3.1 Mining fuzzy emerging patterns

The first step for classification using fuzzy emerging patterns is extracting them from a given training sample. To do that we fuzzify all attributes. For non-numeric attributes, we create a collection of singleton fuzzy sets, i.e. for each different value, we create a fuzzy set having membership 1 for that value, and 0 for the remaining values. For numeric attributes, we apply a simple fuzzification method described in the experimental result section. Next, we extract fuzzy emerging patterns from a set of different fuzzy decision trees induced from the training sample. The induction algorithm is a variant of the ID3 method for the fuzzy case, with the following diferences:

- Candidate splits use the following fuzzy sets and their fuzzy negation:
  1. Every single fuzzy set obtained in the fuzzyfication step
  2. Fuzzy sets in previous item, modified by all different predefined hedges
- In classical ID3, each object in a decision node is assigned to a single child node. In the fuzzy version, each object is assigned to all child nodes with a membership value according to its membership value in the parent node and the fuzzy set associated with the child node. This way, every object belongs to all nodes in the fuzzy tree with a different membership value. To calculate the object membership in a child node, our algorithm applies a fuzzy AND to the object membership in the parent node with the membership of the object to the fuzzy set associated with the child node.

```
Data: T - training sample
Result: ResultFEP
FEP ← ∅;
forall o ∈ T do
    set μ_T(o) = 1
end
for k1 ← 1 to 5 do
    for k2 ← 1 to 4 do
        for k3 ← 1 to 3 do
            for k4 ← 1 to 2 do
                Tree ← BuildNode(T, {μ_T(o)} 1, {k1, k2, k3, k4, 1});
                FEP ← FEP ∪ ExtractPatterns(Tree)
            end
        end
    end
end
ResultEP ← RemoveDuplicates(FEP);
```

Algorithm 1: Algorithm MinePatterns

- In the fuzzy version, we use the following stopping criteria:
  - The node is pure, i.e. all the objects with non-zero membership to the node belong to the same class
  - For all classes, the total membership of the objects to the node is below a given threshold $\mu_{\min}$, that is $\forall C_i \in T : \sum_{o \in C_i} \mu_N(o) \leq \mu_{\min}$, where $\mu_N(o)$ is the membership of the object o to the node N
- The use of *fuzzy information gain* (fig) of a node N. The fig is an extension of the ID3 information gain for the fuzzy case [23]:

$$fig(N) = fimp(N) - \sum_{Nc \in child(N)} fimp(Nc) \cdot \frac{\sum_{o \in Nc} \mu_{Nc}(o)}{\sum_{o \in N} \mu_N(o)} \tag{1}$$

  where $\mu_N(o)$ refers to the membership of the object o to the node N, and the *fuzzy impurity* $fimp(N)$ is defined as:

$$fimp(N) = - \sum_{C \in classes(N)} \frac{\sum_{o \in C} \mu_N(o)}{\sum \mu_N(o)} \cdot \log \left(\frac{\sum_{o \in C} \mu_N(o)}{\sum \mu_N(o)} \right)$$

To guarantee diversity among the trees used to extract the fuzzy emerging patterns, we select a trade-off between the best tree (the tree with the highest fuzzy information gain in all splits) and all possible trees, since the former is unique, and the last is hard to apply to nontrivial problems because of its time complexity. In our algorithm, to build a tree, we expand the best k candidate splits, with a decreasing cardinal according to the node level k_{level}. This allows higher diversity in upper nodes, where there are often more good-splits, reducing the diversity in lower nodes, where there are less good-splits. In our experiments, we evaluate the best 5 splits in the root node, the best 4 in the next level, and so on (5, 4, 3, 2, 1), and we select the best split in lower levels. This way, we create $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$ different trees. A formal description of the pattern mining procedure appears in Algorithm 1.

For example, Table 2 contains the description of three objects, using three attributes: temperature, rain and humidity. Figure 1 shows the membership functions generated using a fixed-bin fuzzy discretization, with three bins.

Data: T – object collection to build the tree, $\{\mu_T(o)\}$ - associated membership of the object o to the current node, l – level in the tree of the resultant node, $\{k_{level}\}$ – set of k values for each level
Result: N – decision node
while *T has objects in more than one class* $\wedge \exists C_i \in T : \sum_{o \in C_i} \mu_T(o) > \mu_{min}$ **do**
  Generate all candidate splits S_i using all fuzzy sets and hedges;
  Calculate $fig(S_i)$, the fuzzy information gain of every split S_i;
  Sort S_i in descending order according to $fig(S_i)$;
  Find S', the ${k_l}^{th}$ element of the sorted S_i collection;
  Construct the left and right child node fuzzy sets FS_{left} and FS_{right} according to S'. FS_{left} corresponds to the fuzzy set associated to S', while FS_{right} corresponds to the negated fuzzy set, using a fuzzy negation operator;
  Find the child node subsets T_{left} and T_{right}, according to the split S'. Calculate $\left\{\mu_{T_{left}}(o)\right\}$ and $\left\{\mu_{T_{right}}(o)\right\}$, as the product of its current value and the membership of the object to FS_{left} and FS_{right} respectively;
  $N_{left} \leftarrow BuildNode\left(T_{left}, \left\{\mu_{T_{left}}(o)\right\}, l+1, \{k_{level}\}\right)$;
  $N_{right} \leftarrow BuildNode\left(T_{right}, \left\{\mu_{T_{right}}(o)\right\}, l+1, \{k_{level}\}\right)$;
  Construct the decision node N, with child nodes N_{left} and N_{right} respectively;
end

Algorithm 2: Algorithm BuildNode

Table 2 Description of the objects used in the example

Object	Class	Temperature (°C)	Rain (mm)	Humidity (%)
o1	Bolded	95	120	5
o3	Bolded	50	10	40
o2	Non-bolded	70	150	15

We generate 120 decision trees, starting from (1, 1, 1, 1, 1) to (5, 4, 3, 2, 1). For example, the tree (2, 1, 3, 2, 1) is built using the second best split in the root node, the best split in the second level, the third best split in the third level, and so on. To expand the root node, we generate all candidate splits (Fig. 2).

We calculate the fuzzy information gain on each candidate split using Equation 1 (Fig. 3), sort the splits according to the fig (Fig. 4), and select the second highest value (Fig. 5).

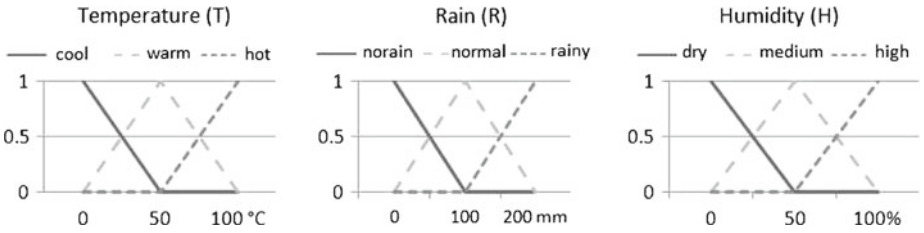


Fig. 1 Fuzzy membership functions per attribute used in the example

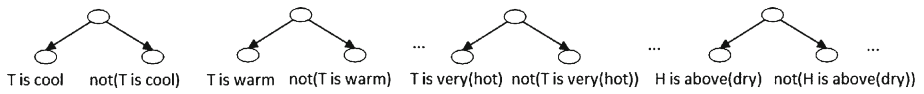


Fig. 2 Candidate splits in the example

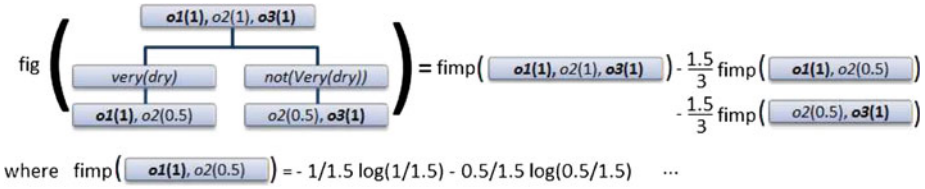


Fig. 3 Evaluating the *fuzzy information gain* (fig) in a candidate split example

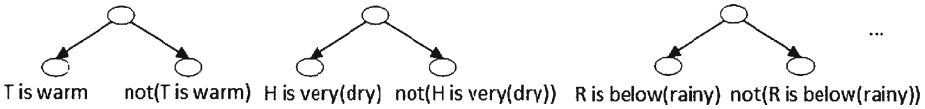


Fig. 4 Sorting candidate splits in the example according to the fig value

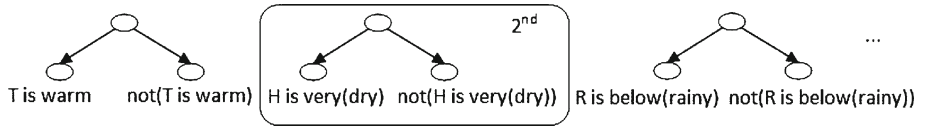
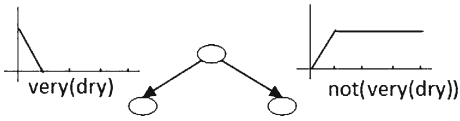


Fig. 5 Selecting the second candidate split in the example

Fig. 6 Creating a new decision node and its related fuzzy sets



Finally, we built a new decision node using the fuzzy sets associated with the selected split (Fig. 6) and recursively apply the whole procedure to each child node until we find a leaf node. Figure 7 shows the complete fuzzy decision tree (2,1,3,2,1) generated by the algorithm BuildNode (Algorithm 2).

From each tree, we extract all the fuzzy emerging patterns, which are the conjunctions of the properties in the paths from the root node to the leaves. Each pattern is assigned to the class with the highest fuzzy support. Finally, we remove duplicated patterns and patterns

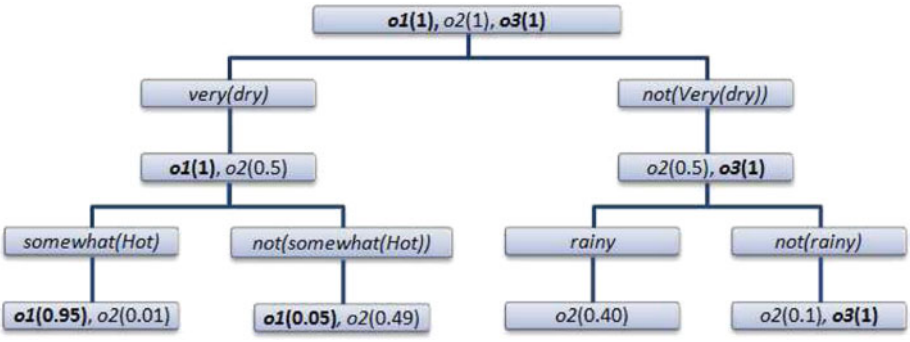


Fig. 7 Fuzzy tree generated in the example. The membership of the objects to each node appears between parenthesis

Table 3 Fuzzy emerging patterns extracted from the tree in Fig. 7

Class	FEP	$Trust$
Bolded	$very(dry) \wedge somewhat(hot)$	0.99
	$not(very(dry)) \wedge not(rainy)$	0.91
Non-bolded	$very(dry) \wedge not(somewhat(hot))$	0.91
	$not(very(dry)) \wedge rainy$	1.00

with a $Trust$ below 0.5, because they have more fuzzy support to the remaining classes than to its own class.

For example, in the tree appearing in Fig. 7 we have four leaves, so we extract the four fuzzy emerging patterns shown in Table 3. In this case, any pattern is discarded because they have $Trust > 0.5$. It is important to highlight the expressivity and easiness to understand the extracted patterns.

3.1.1 Computational complexity

The computational cost of our mining procedure is a constant (in our case 120) multiplied by the cost of inducing each fuzzy decision tree. Extracting the fuzzy emerging patterns does not increase the complexity because the patterns can be directly extracted from the trees during the mining procedure.

The algorithm we use to induce fuzzy decision trees has similar complexity than crisp decision tree induction algorithms; the following are the main differences:

- A fixed amount of splits are used for each numerical attribute, instead of dynamically calculating the cut points as in the crisp version.
- In the crisp version, every object is assigned to a single child, so in the case of balanced trees, you get $\log(n)$ levels. In the fuzzy version, most objects are assigned to both child nodes with different membership values, so the tree has more depth.

Section 4.3 presents an experimental study about the scalability of the mining procedure by increasing the number of objects, attributes, and discretized values.

3.2 Classifying with fuzzy emerging patterns

To understand the classification stage of the classifier, we first introduce the concept of more general pattern and more particular pattern.

Definition 4 Let fep_1 and fep_2 be two fuzzy emerging patterns. fep_1 is more general than fep_2 if for all F-selectors in fep_2, fep_1 contains an equal or more general f-selector. An F-selector $fs_1 \equiv [Attr_1 \in FS_1]$ is more general than another F-selector $fs_2 \equiv [Attr_2 \in FS_2]$ if $Attr_1 = Attr_2$ and $FS_2 \subset FS_1$. Note that fep_2 can have selectors unrelated to any selector in fep_1.

It is important to highlight that if a pattern does not explicitly contain a selector for a particular attribute $Attr$, it contains implicitly the selector $[Attr \in Domain(Attr)]$.

Definition 5 Let fep_1 and fep_2 be two fuzzy emerging patterns. fep_1 is more particular than fep_2 if fep_2 is more general than fep_1.

If two patterns contain linguistic hedges, we use the fuzzy subset inclusion between them to select the more general. This way, the F-selector $very(high)$ is more general than

$extremly(high)$ but less general than $high$. The "more general" relation is antisimetric, so a pattern can be unrelated with other patterns. Also, a more general pattern supports every object with equal or higher degree than a less general pattern.

Particular patterns reduce duplicate pattern contribution [24] and provide more information about relations between attributes [25]. Nevertheless, they are harder to find in a query object, generating abstention [24]. On the other hand, general patterns are more resistant to noise [14,26] and can be mined with less computational effort [14]. Nevertheless, aggregating many minimal patterns may implicitly cause duplicate counting of individual pattern contributions, which could decrease classification accuracy [27,24]. That is why, we propose a novel mixed strategy to build the classifier, which gets the best of both types of patterns. In our classifier, we build a pattern graph as follows:

- Nodes are associated with fuzzy emerging patterns
- Arcs connect nodes with more particular patterns to nodes with more general patterns, according to the antisymmetric relation "is more particular than". If an arc connects two nodes N_1 and N_2 and there is another path between both nodes, the arc is discarded.

In this graph, nodes with no ancestors are maximal patterns (more particular) because any other pattern contains it. Similarly, nodes with no successors are minimal patterns (more general). The algorithm for creating such graph is straightforward. We evaluate every possible pair of nodes, testing for the fulfillment of the relation "is more particular than". If it holds, we create the corresponding arc. Finally, a post-processing step discards longer redundant paths.

To compute the votes per class of a query object, the proposed classifier named FEPC starts evaluating the patterns with no ancestors. If the evaluated pattern matches the query object (with a fuzzy support above a certain threshold), the vote to its class is increased with its $Trust$, while all its successors are discarded. Otherwise, all successors are evaluated in the same way. The process ends when every single node has been evaluated or discarded. Finally, FEPC assigns to the query object the class with the highest total support (See Algorithm 3).

Data: q – query object to classify, G – fuzzy emerging pattern graph
Result: $Classification$ – class assigned by the classifier to the query object

$Processed \leftarrow \emptyset$;
foreach *class* C **do**
  $Votes_C \leftarrow 0$
end
$Pendant \leftarrow$ FEPs with no ancestors in G ;
foreach $fep \in Pendant$ **do**
  $Processed \leftarrow Processed \cup \{fep\}$;
  if fep *match* q **then**
    $Votes_{class(fep)} \leftarrow Votes_{class(fep)} + Trust(fep)$;
    $Processed \leftarrow Processed \cup \{$ descendants of fep in $G\}$
  else
    $Pendant \leftarrow Pendant \cup \{$ descendants of fep in $G\}$
  end
  if $\exists Class : Votes_{Class} > 0$ **then**
    $Classification \leftarrow \arg\max_C \{Votes_C\}$
  else
    $Classification \leftarrow \emptyset$
  end
end

Algorithm 3: Algorithm FEPC, Fuzzy Emerging Pattern Classifier

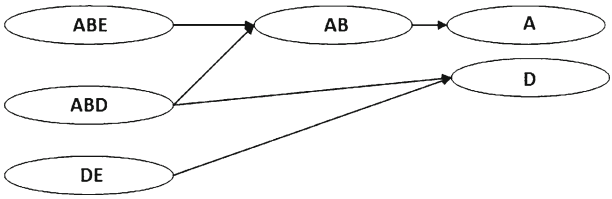


Fig. 8 Example of pattern tree built by FEPC

As an example of the classification algorithm, suppose we have the following patterns **ABE**, **ABD**, **DE**, **AB**, **A**, and **B**, formed by the selectors **A**, **B**, **C**, **D** and **E**. Using the more general relationship, FEPC builds the pattern graph appearing in Fig. 8. If we want to classify the object **ABE**, FEPC starts considering the patterns with no ancestors. Like any of them matches the query object, it considers their successors **AB** and **D**. Since there is a match with **AB**, the pattern **A** is discarded. Finally, the pattern **AB** is the only one considered for classification. In a similar way, to classify the object **BDE**, FEPC classifier considers only the pattern **DE**, while to classify the object **ADE**, it considers the patterns **DE** and **A**.

FEPC tries to classify each object with fuzzy emerging patterns as particular as possible. If a pattern does not match with the query object, the classifier considers more general patterns, using the arcs in the graph. With this strategy, we can combine the particular pattern lower errors, with the general pattern low abstention in a synergic way. This way, FEPC strategy avoids vote duplication, which frequently leads to incorrect classification in problems where there are many patterns very similar to each other.

4 Experimental results

4.1 Experimental setup

To compare the performance of the proposed method, we carried out some experiments over 16 databases from the UCI repository of Machine Learning [17]. The description of the tested databases appears in Table 4.

To make comparisons, we selected seven popular and state of the art classifiers. They are k Nearest Neighbors [18] with $k = 3$ and 7, Boosting and Bagging [28], Random Forest [29], C4.5 [30] and Support Vector Machines [31]. We use the Weka [32] implementation with the default parameters for each method. We include one of the most accurate emerging pattern classifier, SJEP [10], using the threshold value suggested by its authors. In both SJEP and FEPC, we report the objects with no supporting patterns as errors. In these objects, the classifier is unable to assign a class and assigning the majority or random class can hide this undesirable behavior.

For our method (FEPC), we construct for each numerical attribute a Ruspini fuzzy partition. We create d uniformly distributed fuzzy sets, the first and last with a trapezoidal shape, and inner sets with triangular shapes (like the examples in Fig. 1), which is a commonly used configuration in mining tasks [22,33]. We experimentally determine that the highest accuracy appears using $d = 4$, but values 3 and 5 achieve similar accuracy results. We use $\mu_{\min} = 0.05$ for all threshold values, although we did not find statistically significant differences in the accuracies using values between 0.05 and 0.15. Also, we experimentally found

Table 4 Database description

Database	# of attributes		Objects	Distribution per class
	Non-numeric	Numeric		
Autos	10	15	205	13/26/16/33/11/1
Breast-canc-wisc	0	9	699	66/34
Credit-screening	9	6	690	44/56
Cylinder-bands	21	18	540	42/58
Glass	0	9	214	33/8/4/36/14/6
Hepatitis	13	6	155	79/21
Ionosphere	0	34	351	64/36
Iris	0	4	150	33/33/33
Labor	8	8	57	65/35
Lymph	15	3	148	41/3/55/1
mp1	6	0	556	50/50
mp2	6	0	601	66/34
mp3	6	0	554	52/48
Tic-tac-toe	9	0	958	65/35
wdbc	0	30	569	37/63
wpbc	0	33	198	76/24

that the shape of the fuzzy set has a low impact in the quality of the mined patterns, compared to other similar shapes (Sigmoid, beta) used in knowledge mining tasks. Nevertheless, the number of bins has a significant negative impact in the algorithm efficiency because using more bins implies evaluating more candidate splits on each iteration of the tree-building procedure.

4.2 Accuracy comparisons

In Table 5, we show the accuracy results, in percent, of the experiments. We perform 10-fold cross validation, averaging the results. It is easy to notice that our classifier outperforms all the tested classifiers in most of the databases.The poor results of SJEP in databases *autos* and *glass* are due to the poor results of its inner discretization step. In database *wpbc*, SJEP was unable to extract even a single pattern because most numerical attributes were converted into a non-numerical attribute with a single value.

In order to determine if the differences in accuracy are statistically significant, we performed a pairwise comparison between our classifier and the others. Each cell in Table 6 contains the number of databases where our classifier Win/Lose/Tie to each other classifier. We detected ties using a two-tailed T-Test [34] with significance of 0.05. The results in the pairwise comparison reveal that our classifier beats in accuracy all other single classifiers in almost all the tested databases.

Tables 5 and 6 show that the new classifier performs better than the others in many but not all cases. Although FEPC clearly outperforms other non-metric methods (Bagging, Boosting, c4.5, Random Forest and SJEP), it frequently ties with metric methods (kNN and SVM). To explain this behavior, we should point out that our patterns are unable to capture relations among attributes, which are easily captured by distance-based methods.

Table 5 Accuracy of the tested classifiers on the selected databases

Database	3 nn	7 nn	Boost	Bagg	RFor	c4.5	SVM	SJEP	FEPC
Autos	68.24	59.88	42.40	61.12	81.00	80.95	66.00	12.31	**85.41**
Breast-cancer-w	96.53	95.71	95.58	95.57	96.47	95.93	96.99	96.28	**98.79**
Credit-screening	84.15	86.24	86.28	85.94	85.02	85.08	86.30	82.61	**88.94**
Cylinder-band	70.16	71.19	72.86	60.37	78.51	72.19	80.89	64.28	**89.83**
Glass	69.03	62.03	44.74	**73.77**	70.93	67.71	57.08	20.43	59.71
Hepatitis	**86.04**	85.08	83.58	82.00	82.37	81.79	85.16	83.21	85.84
Ionosphere	85.47	83.75	91.44	90.29	92.58	90.30	88.04	**93.53**	93.37
Iris	96.59	97.21	**97.75**	95.83	95.20	95.83	97.04	75.33	97.67
labor	**90.67**	89.33	87.00	84.00	86.67	80.00	**90.67**	55.33	90.33
Lymph	85.90	83.04	75.67	77.67	79.86	78.48	87.86	43.86	**91.96**
mp1	81.02	76.85	75.00	50.00	75.69	88.89	50.00	86.81	**99.07**
mp2	61.34	65.28	60.65	55.09	65.05	69.88	50.46	71.06	**75.00**
mp3	88.19	92.82	**97.22**	50.00	**97.22**	96.30	50.00	93.52	96.06
Tic-tac-toe	97.89	97.89	74.73	85.26	92.63	85.26	95.79	98.80	**99.44**
wdbc	96.27	95.94	92.32	94.38	94.02	91.43	**97.71**	85.07	96.41
wpbc	72.30	75.95	71.02	78.78	74.69	75.51	75.88	0	**83.38**

The highest accuracy result for each database appears bolded

Table 6 Pairwise comparison between our classifier and the others

	3nn	7nn	Boost	Bagg	RFor	c4.5	SVM	SJEP
FEPC	10/1/5	10/1/5	13/0/3	14/1/1	13/1/2	13/1/2	11/0/5	14/0/2

Each cell shows the number of Win/Loss/Tie of FEPC with respect to the corresponding classifier over the selected 16 databases

4.3 Algorithm scalability

In order to test the scalability of our mining algorithm by increasing the number of objects, the number of attributes, and the number of discretized values of the attributes, we use the database *hypothyroid*. *Hypothyroid* has 3772 objects, 7 numerical attributes, and 22 non-numerical attributes. We evaluated the impact of increasing each parameter, testing different values while keeping the other parameters unaltered. The results were the following:

Number of objects Adding new objects has different levels of impact, depending on characteristics of the objects. An object similar to previous objects in its same class usually does not alter the decision tree. An object very dissimilar to previous objects, or similar to objects in a different class, could force the mining procedure to make more splits. Figure 9 shows a linear dependency between the number of objects and the time needed to mine the patterns.

Number of attributes Adding a new attribute increments the number of candidate splits in every node of the tree by $k \cdot NumHedges$, where $NumHedges$ is the amount of hedges considered in the system, and k is the number of values of the attribute (4 for numeric attributes). Figure 9 shows an exponential dependency on the number of attributes.

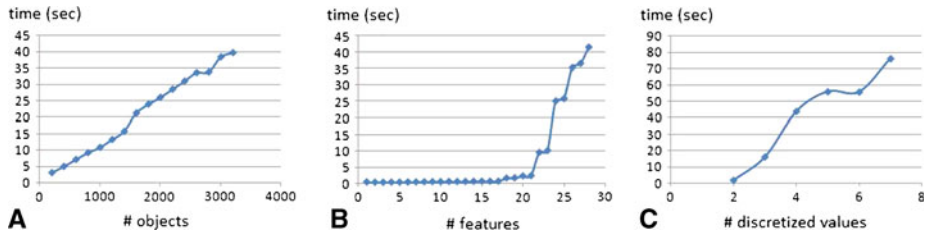


Fig. 9 Scalability results of FEP miner by increasing the number of objects (**a**), attributes (**b**), and values of the attributes (**c**)

Number of discretized values Increasing the number of discretized values adds $Num\ Hedges$ candidate splits in every node of the tree. Figure 9 shows a dependency close to linear.

Generally speaking, our algorithm scales better to adding new objects than to adding new discretized values, and scale worst to adding new attributes.

5 Conclusions

In this paper, we introduce fuzzy emerging patterns, an emerging pattern extension to the fuzzy case, aiming at a better discretization of continuous attributes. In order to extract these fuzzy patterns from a training sample, we propose a new procedure based on the induction of several fuzzy decision trees. The induction procedure uses linguistic hedges to fix the initial fuzzy discretization of the continuous attributes, which is also a contribution of this paper. We also introduce a measure to test the discriminability of a fuzzy emerging pattern for classification, named *Trust*.

Using the extracted fuzzy emerging patterns, we propose the classifier FEPC, which uses a novel graph-based strategy for organizing the patterns. This strategy allows to create a more accurate classifier, with lower levels of abstention. In our experiments, the FEPC classifier shows significant higher levels of accuracy than some popular and state of the art classifiers. In the pairwise comparison, FEPC beats every other single classifier in the majority of the tested databases.

In order to improve some of the limitations of our algorithm, we are going to study the following approaches as future work:

- Inclusion of splits containing more than one attribute in the tree induction algorithm in order to capture relations among attributes. This is a complex task because it could significantly degrade the mining efficiency because many more splits have to be considered in every node. Additionally, finding the splitting hyperplanes can be a time-consuming task.
- Selection of a dynamic number of bins for the fuzzy discretization, according to the attribute value distribution.
- Generation of a variable number of fuzzy trees, according to the complexity of the database.
- Using the patterns mined in previous decision trees in order to guide the construction of the following trees. This could prevent the same patterns to appear in many different trees, with the consequent waste of time.

Acknowledgments The authors want to thank the anonymous reviewers for their valuable suggestions, which significantly improved the quality of this paper. This work is partly supported by The National Council of Science and Technology of Mexico under the project CB-2008-01-106443 and grant 25275.

References

1. Bongard MN (1963) Solution to geological problems with support of recognition programs. Sov Geologia 6:33–50
2. Keilis-Borok A, Soloviov A (1991) Pattern recognition: general description. In: Workshop in non-linear dynamics and earthquake prediction, International Center for Science and High Technology, Trieste, Italy, pp 1–14
3. Michalski RS, Stepp R (1982) Revealing conceptual structure in data by inductive inference. In: Michie D, Hayes JE, Pao HH (eds) Machine Intelligence, vol. 10. Ellis Horwood Ltd, New York, pp 173–196
4. Dong G, Li J (1999) Efficient mining of emerging patterns: discovering trends and differences. In: Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining, San Diego, California, United States, ACM, pp 43–52
5. Ramamohanarao K, Fan H (2007) Patterns based classifiers. World Wide Web 10(1):71–83
6. Li J, Dong G, Ramamohanarao K (2000) Instance-based classification by emerging patterns. In: Proceedings of the 4th European conference on principles of data mining and knowledge discovery. Springer, pp 191–200
7. Dong G, Zhang X, Wong L, Li J (1999) Caep: classification by aggregating emerging patterns. In: DS'99, vol. 1721 of Lecture Notes in Computer Science, Japan
8. Hämälïnen W (2009) Statapriori: an efficient algorithm for searching statistically significant association rules. Knowl Inf Syst. doi:10.1007/s10115-009-0229-8
9. Jin R, Breitbart Y, Muoh C (2009) Data discretization unification. Knowl Inf Syst 19:1–29
10. Fan H, Ramamohanarao K (2006) Fast discovery and the generalization of strong jumping emerging patterns for building compact and accurate classifiers. IEEE Trans Knowl Data Eng 18(6):721–737
11. Zadeh LA (1965) Fuzzy sets. Inf Control 8:338–353
12. Weng C-H, Chen Y-L (2009) Mining fuzzy association rules from uncertain data. Knowl Inf Syst. doi:10.1007/s10115-009-0223-1
13. González A, Pérez R (1999) A study about the inclusion of linguistic hedges in a fuzzy rule learning algorithm. Int J uncertain fuzziness knowl based syst 7(3):257–266
14. Fan H, Ramamohanarao K (2002) An efficient single-scan algorithm for mining essential jumping emerging patterns for classification. In: Proceedings of the 6th Pacific-Asia conference on advances in knowledge discovery and data mining, Springer, pp 456–462
15. Fayyad U, Irani K (1993) Multi-interval discretization of continuous-valued attributes for classification learning. In: 13th Int'l Joint Conf Artif Intell (IJCAI), pp 1022–1029
16. Appice A, Ceci M, Malgieri C, Malerba D (2007) Discovering relational emerging patterns. In: AI*IA 2007: artificial intelligence and human-oriented computing, pp 206–217
17. Merz C, Murphy P (1998) Uci repository of machine learning databases. Technical Report, University of California at Irvine, Department of Information and Computer Science
18. Dasarathy BD (1991) Nearest Neighbor (NN) norms: NN pattern classification techniques. IEEE Computer Society Press, Los Alamitos California
19. Yuan Y, Shaw M (1995) Induction of fuzzy decision trees. Fuzzy Sets Syst 69:125–139
20. Wang XZ, Chen B, Qian G, Ye F (2000) On the optimization of fuzzy decision trees. Fuzzy Sets Syst 112:117–125
21. Wang XZ, Zhai JH, Zhang SF (2008) Fuzzy decision tree basen on the important degree of fuzzy attribute. In: 2008 international conference on machine learning and cybernetics, vol. 1, Kunming, pp 511–516
22. Huang D-M (2008) An algorithm for generating fuzzy decision tree with trapezoid fuzzy number-value attributes. In: International conference on wavelet analysis and pattern recognition, ICWAPR 08, vol. 1, Hong Kong, China, pp 41–45
23. Dong M, Kothari R (2001) Look-ahead based fuzzy decision tree induction. IEEE Trans Fuzzy Syst 9(3):461–468
24. Wang Z, Fan H, Ramamohanarao K (2004) Exploiting maximal emerging patterns for classification. In: 17th Australian joint conference on artificial intelligence, Cairns, Queensland, Australia, pp 1062–1068

25. Zhang X, Dong G, Ramamohanarao K (2000) Information-based classification by aggregating emerging patterns. In: Proceedings of the second international conference on intelligent data engineering and automated learning, data mining, financial engineering, and intelligent agents, Springer, pp 48–53
26. Fan H, Ramamohanarao K (2003) A bayesian approach to use emerging patterns for classification. In: Proceedings of the 14th Australasian database conference—Volume 17, Adelaide, Australia, pp 39–48, Australian Computer Society, Inc
27. Bailey J, Manoukian T, Ramamohanarao K (2002) Fast algorithms for mining emerging patterns. In: Proceedings of the 6th European conference on principles of data mining and knowledge discovery, vol. 2431 of Lecture Notes in Computer Sciences, pp 187–208, Springer, 756628 39–50
28. Kuncheva LI. Combining pattern classifiers. Methods and algorithms. Wiley-Interscience, Hoboken
29. Ho TK (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20(8):832–844
30. Quinlan JR (1993) C4.5: Programs for machine learning. Morgan Kaufmann Publishers Inc, San Francisco
31. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
32. Wittn I, Frank E, Trigg L, Hall M, Holmes G, Cunnigham S (1999) Weka: practical machine learning tools and techniques with java implementations. In: Emerging knowledge engineering and connectionist-based information systems, pp 192–196
33. Alcalá-Fdez J, Alcalá R, Gacto MJ, Herrera F (2009) Learning the membership function contexts for mining fuzzy association rules by using genetic algorithms. Fuzzy Sets Syst 160(7):905–921
34. Dietterich TG (1998) Approximate statistical tests for comparing supervised classification learning algorithms, vol. 10. MIT Press, Cambridge

Author Biographies

Milton García-Borroto graduated from Las Villas Central University, Cuba, in 2000. He received his M.Sc. degree in 2007 from the National Institute of Astrophisics, Optics and Electronics, Mexico, where he continues his studies toward a Ph.D. degree. His research interests are pattern recognition, intelligent systems, machine learning, and biometry.

José Fco Martínez-Trinidad received his B.S. and M.Sc. degrees in Computer Science from the Autonomous University of Puebla, Mexico in 1995 and 1997, respectively and his Ph.D. degree in the National Polytechnic Institute, Mexico in 2000. Professor Martinez-Trinidad edited/authored four books and over fifty papers on subjects related to Pattern Recognition.

Jesús Ariel Carrasco-Ochoa received his Ph.D. in Computer Science from the Center for Computing Research of the National Polytechnic Institute, Mexico, in 2001. He works as full-time researcher at the National Institute for Astrophysics, Optics and Electronics, Mexico. His current research interests include:Pattern Recognition, Feature and Prototype Selection, and Clustering.