Fuzzy Partition Inference Over a Set of Numerical Values

Christophe Marsala

LAFORIA-IBP,

UNIVERSITÉ PIERRE ET MARIE CURIE

Case 169, 4 place Jussieu, 75252 Paris Cedex 05, France.

e-mail: marsala@laforia.ibp.fr

October 1995

Abstract

In this paper, we present an algorithm to infer a fuzzy partition over a set of numerical values. This algorithm is based on the mathematical morphology and is expressed in the formal language theory. We use it during the construction of a fuzzy decision tree, in the case where no fuzzy partition is available for a numerical attribute.

Résumé

Dans ce rapport, nous présentons un algorithme pour construire une partition floue sur un ensemble de valeurs numériques. Cet algorithme s'inspire des opérateurs de la morphologie mathématique pour définir des zones qui serviront de base à des sous-ensembles flous. Nous formalisons cette approche à l'aide de la théorie des langages, sous la forme de systèmes de réécriture.

Nous utilisons cet algorithme durant la construction d'un arbre de décision flou, pour trouver une partition floue pour un attribut à valeurs numériques, quand aucune connaissance préalable sur cet attribut n'est fournie.

Introduction

In inductive learning, and particularly in induction of decision trees, a lot of methods do not fit very well the numerical nature of the data to deal with.

To build a decision tree, some existing methods adapt the popular Quinlan's ID3 algorithm [Quinlan86, Quinlan93] to take into account the problem of such training sets of data. A first kind of methods considers numerical data as symbolic ones [Quinlan86]. Then, some fuzzy-oriented tools can be introduced to use the built tree [Maher et al.93]. However, this kind of methods produces large trees and we can consider that it is difficult to rely on induction from a single numerical datum. Other methods introduce a partitioning step where we search for cut points on a numerical universe, either before [Carter et al.87, Catlett91] or during the construction of the tree [Breiman et al.84]. But for values near the cut points, the generated tests of the tree are often very imprecise in nature. Therefore, one possible way of improvement lies in the use of fuzzy concepts when classifying a new case, in order to smooth the boundaries found during partitioning step [Jang94, Marsala94, BM et al.95]. A last kind of methods proposes the use of a new information measure during the learning phase, for instance Ramdani's system [Ramdani92, Ramdani94], Fayyad's system [Fayyad et al.93] or Janikow's system [Janikow94]. Ramdani's method is concerned with the construction of decision trees when expert knowledge regarding the data is available. This expert knowledge provides a fuzzy partition upon the universe of the data. Ramdani introduces a new information measure, the *entropy-star*, to take into account this additional knowledge during the construction of the decision tree. The entropy-star is a powerful tool that generalizes the concept of Shannon entropy [Shannon et al.49].

Our purpose is to use the Ramdani's version of the ID3 algorithm when such expert knowledge is not available. However, in order to use the entropy-star, we have to infer a fuzzy partition upon the universe of the numerical data. This fuzzy partition must be coherent, and must replace a fuzzy partition obtained from an expert.

In this paper, we present the solution we have implemented in our system. This solution is based on the use of mathematical morphology operators, and it is formalized by means of the formal language theory.

In Section 1, we recall some elementary basis of formal language theory and some notions of mathematical morphology. In Section 2, we present our rewriting systems that implement the mathematical morphology operators to smooth a word. In Section 3, we propose an algorithm to infer a fuzzy partition upon a set of numerical values. And, finally, we conclude on new developments it will be interesting to add to our method.

1 Preliminaries

1.1 On the formal language theory

In this section, we recall some elementary basis of the formal language theory. For more developments, see [Ginsburg66] or [Autebert87].

Definition 1 *An* alphabet *is a finite non-empty set of elements.*

Let A be an alphabet. A *word* on A is a finite sequence $(x_1, x_2, \ldots, x_n)$ of letters in A. We simplify $(x_1, x_2, \ldots, x_n)$ with $x_1 x_2 \ldots x_n$. We distinguish the *empty* word $()$ noted ε. The set of words on A is denoted by A^*. By definition, $\varepsilon \in A^*$.

Let v, w be two words from A^*.

Definition 2 *The* concatenation *of v and w is the word $v.w \in A^*$ such that $v = x_1 x_2 \ldots x_n$, $w = y_1 y_2 \ldots y_p$ and $v.w = x_1 x_2 \ldots x_n y_1 y_2 \ldots y_p$*

The concatenation is an *internal relation* that admits the word ε as neutral element. Usually, $v.w$ is noted vw.

We note x^n a sequence of n successive identical letters x in a word: $\underbrace{x \ldots x}_{n} = x^n$. A sequence x^n in a word v is called *complete* if it cannot be enlarged in v. For example, in $v = abb^3ac^4a$, b^3 is not a complete sequence and c^4 is a complete sequence.

Definition 3 *A* rewriting system *is a finite set of ordered pairs of words on a given alphabet.*

For example, with $A = \{a, b, c\}$, $\{(aa, a), (b, bb), (ac, b), (a, \varepsilon)\}$ defines a rewriting system. Each pair (u, v) is a *rewriting rule* (or *production*). Such systems are often noted:

Example of rewriting system:

$$\begin{array}{lclclcl} aa & \longrightarrow & a & \qquad\qquad & b & \longrightarrow & bb \\ ac & \longrightarrow & b & & a & \longrightarrow & \varepsilon \end{array}$$

We say that v is *rewritten* in w if there exists a sequence of rewriting rules from v to w. We note $v \xrightarrow{*} w$. The * upon the $\longrightarrow$ denotes the use of zero or more rewriting rules. w is *irreducible* if it cannot be rewritten in another word. v possesses an irreducible if there exists an irreducible w such that $v \xrightarrow{*} w$.

Definition 4 *A rewriting system is* nœtherian *if there does not exist any infinite sequence of rewriting rules.*

For example, the above rewriting system is not nœtherian because the word b admit an infinite sequence of rewriting rules.

Definition 5 *A rewriting system is* confluent *if for all 3-tuples of words u, v and w such that $u \xrightarrow{*} v$ and $u \xrightarrow{*} w$, there exists a word t such that $v \xrightarrow{*} t$ and $w \xrightarrow{*} t$.*

Definition 6 *A rewriting system is called* local-confluent *if for all 3-tuples of words u, v and w such that $u \longrightarrow v$ and $u \longrightarrow w$, there exists a word t such that $v \xrightarrow{*} t$ and $w \xrightarrow{*} t$.*

Theorem 1 *With a nœtherian system, all words possess an irreducible.*

Theorem 2 *With a confluent system, no word possesses more than one irreducible.*

Theorem 3 *A rewriting system that is nœtherian and local-confluent is confluent.*

The proofs for these three theorems are easily found [Autebert87].

It is very important to know if a rewriting system is confluent because an algorithm which implements a confluent rewriting system terminates for any word, and, also, the irreducible of a word is unique.

1.2 On the mathematical morphology

In this section, we present the fundamental operators of the mathematical morphology. The theoretical statements of these operators can be found in [Serra82] or [Coster et al.89].

We consider a space of morphological bodies.

1.2.1 The basic operators: erosion and dilatation

These two basic operators enable the modification of a morphological body $\mathcal{C}$. This modification is related to a *structuring element* $\mathcal{E}$. The *erosion* is a particular substraction of $\mathcal{E}$ in $\mathcal{C}$, and the *dilatation* is a particular addition of $\mathcal{E}$ in $\mathcal{C}$.

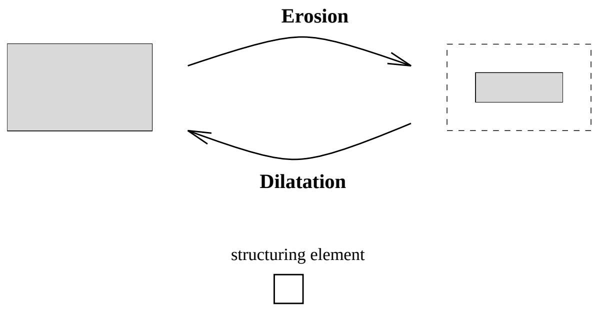

Figure 1: Morpho-maths operator

1.2.2 The operators opening and closure

Each of these operators is a combination of the two basic operators.

The opening is the combination of an erosion followed by a dilatation applied to a morphological body, with the same structuring element. It enables destruction of small bodies in the space, with respect to the size of the chosen structuring element.

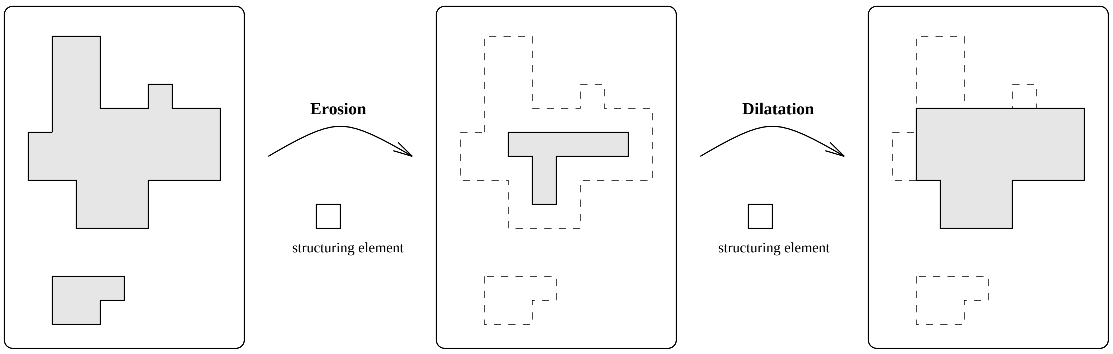

Figure 2: Opening operator

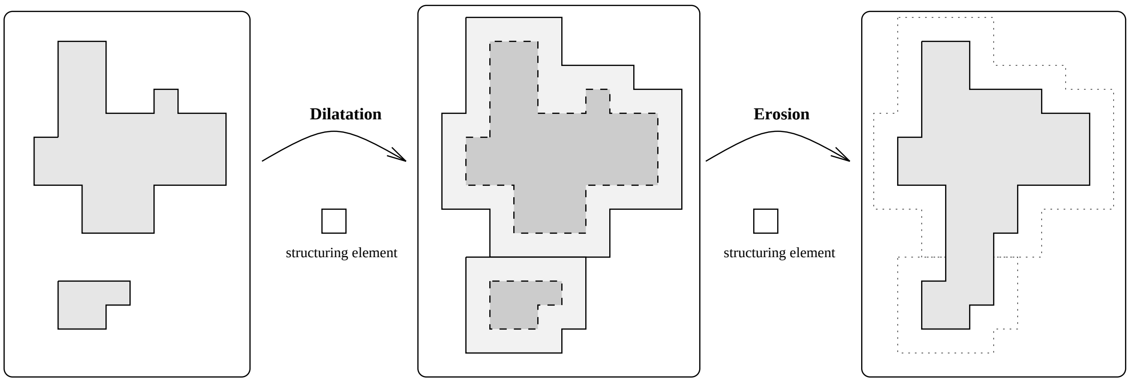

Figure 3: Closure operator

The closure is the combination of a dilatation followed by an erosion applied to a morphological body, with the same structuring element. It enables the destruction of small vacuum places occurring in a body, with respect to the size of the chosen structuring element.

1.2.3 The open-close filter

A filter is a combination of openings and closures. It is composed by n successive openings followed by n successive closures ($n \in \mathbb{N}$) applied to all bodies of the space, with the same structuring element. Thus it enables the destruction of small bodies present in the space with respect to the chosen structuring element. Simultaneously, it enables the filling of small vacuum places occurring in bodies. The value of n enables the control of such destructions.

2 To smooth a word

In this section, we propose a rewriting system to smooth a word, and we prove that this system is confluent.

2.1 The training set as a word

In this part, we show how to transform a training set into a word. We consider the case where the classes we want to learn about are crisp.

Let X_j be the continuous universe of a numerical attribute. In the training set, the values of this attribute is a discrete subset of this universe. We name X_j^d this finite set of values.

Each element of the finite set of classes can be associated with a particular letter from a given alphabet, with the restriction that two distinct elements are associated with two different letters. Let $\mathcal{L}$ be the set of such letters. We construct the alphabets $\mathcal{L}_u = \mathcal{L} \cup \{u\}$, $\Sigma = \mathcal{L}_u \cup \{\alpha, \#, \$\}$ with $u \notin \mathcal{L}, \alpha \notin \mathcal{L}_u, \# \notin \mathcal{L}_u$ and $\$ \notin \mathcal{L}_u$. The letters

$\alpha, \#, \$$ and u are four particular letters we distinguish for our system. In the rewriting system, α will be used to simulate a reading head, # will match the beginning of the word, \$ will match the end of the word and u will highlight particular letters, we will use it to determine *uncertain* places.

With X_j^d, we can build a word in $\mathcal{L}^*$ with the following algorithm:

Algorithm 1 *To translate a training set into a word in $\mathcal{L}^*$:*

1. *sort X_j^d according to the natural order of numbers.*
2. *for each element e of the ordered X_j^d, associate the letter of $\mathcal{L}$ corresponding to its class.*

An element e of X_j^d could be represented more than one time in the training set because several examples could have the same value for a given attribute. Therefore, we add a letter each time e is represented. Moreover, a unique element of X_j^d could appear with different classes in the training set. Thus, a training set could have various translations. A solution is to choose arbitrarily an order upon the classes. Finally, we translate the word $v \in \mathcal{L}^*$ obtained with the *algorithm 1* into a word in Σ^*: $\alpha\#v\$$.

For example, let the training set be {(20, high), (25, low), (35, low), (40, low), (29, low)} with {low, high} as set of classes and $\{20, 25, 29, 35, 40\}$ as set of values of an attribute (e.g. the attribute "age"). Let $\mathcal{L} = \{h, l\}$ be the alphabet associated with the set of classes; *high* is associated with h and *low* is associated with l. The word defined in $\mathcal{L}^*$ by the training set is $hllll$, and the word defined in Σ^* is $\alpha\#hllll\$$.

After this algorithm which enables the translation of the training set into a word, we define various ways of using this word.

2.2 Different operators to smooth a training set

In this part, we present several techniques to alter a word. We want to erase non-representative values within a word in order to smooth it. To define our prime algorithm, we need to introduce some intermediate operators. Such operators are inspired by the mathematical morphology operators.

a. Erosion

Let us define a rewriting system Er_x on Σ^*. This rewriting system is used to *erode* a word with x as structuring element. It corresponds to the reduction of sequences of x in the word.

Er_x : Erosion with respect to x:

$$\begin{array}{rcl@{\qquad\qquad}rcr}
\alpha\#x & \longrightarrow & u\alpha & \alpha\#y & \longrightarrow & \alpha y \\
\alpha\#\$ & \longrightarrow & \alpha\$ & \alpha xx & \longrightarrow & x\alpha x \\
\alpha xy & \longrightarrow & u\alpha y & \alpha x\$ & \longrightarrow & u\alpha\$ \\
\alpha yx & \longrightarrow & yu\alpha & \alpha yz & \longrightarrow & y\alpha z \\
\alpha y\$ & \longrightarrow & y\alpha\$ & \alpha\$ & \longrightarrow & \varepsilon
\end{array}$$

With $x \in \mathcal{L}$ and $y, z \in \mathcal{L}_u$ $y \neq x, z \neq x$. From the word $\alpha\#v\$$, $v \in \mathcal{L}_u^*$, we obtain the word $Er_x(\alpha\#v\$) \in \Sigma^*$, such that $Er_x(\alpha v\$) = w$ with $w \in \mathcal{L}_u^*$.

For example, with the word $v = hllll$, we have $Er_h(\alpha\#hllll\$) = ullll$ and $Er_l(\alpha\#hllll\$) = hullu$.

Now, we have to be sure that our rewriting system is correct. We consider that it will be correct if it stops for any given word and if the computed terminal word is unique.

Theorem 4 *For any word v in $\mathcal{L}_u^*$, for any letter x in $\mathcal{L}$, the eroded word $Er_x(\alpha\#v\$)$ obtained from v exists and is unique*

To prove this theorem, we have to prove that the rewriting system is confluent. It is proved with some classical techniques from the formal language theory: we prove that the theorem is nœtherian and local confluent.

Proof of theorem 4

∘ First, we prove that Er_x is nœtherian.

Let $v_n \in \mathcal{L}_u^*$ be a word with n letters. If $n = 0$ then there does not exist any infinite sequence of rewriting rules from $\alpha\#v_n\$$. The unique productions we can apply are $\alpha\#\$ \longrightarrow \alpha\$$ and $\alpha\$ \longrightarrow \varepsilon$. After that, no production can be applied again.

If $n \neq 0$, the first production used from $\alpha\#v_n\$$ is either $\alpha\#x \longrightarrow u\alpha$ or $\alpha\#y \longrightarrow \alpha y$. We have either $v_n = xv_{n-1}$ or $v_n = yv_{n-1}$. So, the reduction used are either $\alpha\#xv_{n-1}\$ \longrightarrow u\alpha v_{n-1}\$$ or $\alpha\#yv_{n-1}\$ \longrightarrow \alpha yv_{n-1}\$$. In the last case, if $v_{n-1} = xv_{n-2}$ we have $\alpha yxv_{n-2}\$ \longrightarrow yu\alpha v_{n-2}\$$, if $v_{n-1} = yv_{n-2}$ we have $\alpha yyv_{n-2}\$ \longrightarrow y\alpha yv_{n-2}\$$ and if $v_{n-1} = \$$ we have $\alpha y\$ \longrightarrow y\alpha\$$.

Thus, in all cases, at each step, the system reduce the length of the part of the word that could be reduced. For this reason, we deduce that there does not exist any infinite sequence of rewriting rules from $\alpha\#v_n\$ \; \forall n$. Therefore Er_x is nœtherian. A consequence is that the existence of $Er_x(\alpha\#v\$)$ becomes evident (theorem 1).

∘ Now, we prove that Er_x is local-confluent.

For any word in $\mathcal{L}_u^*$, there does not exist two different derivations in one step for this word. It is obvious when we examine all the derivation rules. To apply two different rules on the same word, the first letter after α must be the same for the two rules, and the second letter after α too. It does not exist such two rules in our system; the succession of reduction rules from a single word is unique. Then Er_x is local-confluent.

∘ With theorem 3, we infer that Er_x is confluent, and then, from theorem 2, no word possesses more than one irreducible.

By the way, our theorem is proved.

◇

With this theorem, we are sure that $Er_x(\alpha\#v\$)$ exists for all word $v \in \mathcal{L}_u^*$, and therefore, for all training set.

Now, we present a useful definition.

Definition 7 *For all word $v \in \mathcal{L}_u^*$, we call $Er_x^n(\alpha\#v\$)$, with $n > 0$, the word w, $w \in \mathcal{L}_u^*$, such that:*

$\exists v_0, v_1, \ldots, v_n \in \mathcal{L}_u^*$, $v = v_0$, $v_i = Er_x(\alpha\#v_{i-1}\$) \; \forall i = 1 \ldots n$ *and* $w = v_n$.

$Er_x^n(\alpha\#v\$)$ is obtained from v after n applications in a row of the rewriting system.

With the following theorem, we are able to foresee what kind of modifications this rewriting system produces. Thus, this theorem ensures that the rewriting system is close to an erosion in the mathematical morphology meaning.

Theorem 5 *Let v be a word in $\mathcal{L}_u^*$.*

A complete sequence x^m in v becomes in $Er_x^n(\alpha\#v\$)$ either a sequence u^m if $m \leq 2n$ or a sequence $u^n x^{m-2n} u^n$ if $m > 2n$.

Proof of theorem 5

$\circ$ First, we will prove the theorem for x^m, $m \in \mathbb{N}$.

Let $n = 1$, we have $Er_x(\alpha\#x\$) = u$ and $Er_x(\alpha\#xx\$) = uu$.

If $m > 2$, $Er_x(\alpha\#x^m\$) = ux^{m-2}u$. The applied rules are $\alpha\#x \longrightarrow u\alpha$, $\alpha xx \longrightarrow x\alpha x$ and $\alpha x\$ \longrightarrow u\alpha\$$. Then, the theorem is true for $n = 1$.

Let the theorem be true for a rank n.

Let $m \in \mathbb{N}$, if $m \leq 2n$, $Er_x^n(\alpha\#x^m\$) = u^m$ and $Er_x(\alpha\#u^m\$) = u^m$ then $Er_x^{n+1}(\alpha\#x^m\$) = u^m$.

If $m > 2n$, $Er_x^n(\alpha\#x^m\$) = u^n x^{m-2n} u^n$, and if $m - 2n > 2$, $Er_x(\alpha u^n x^{m-2n} u^n\$) = u^n u x^{m-2n-2} u u^n$. If $m - 2n = 1$, $Er_x(\alpha u^n x u^n\$) = u^n u u^n$. And if $m - 2n = 2$, $Er_x(\alpha u^n xx u^n\$) = u^n uu u^n$.

Therefore, if $m \leq 2(n+1)$, $Er_x^{n+1}(\alpha\#x^m\$) = u^m$ and if $m > 2(n+1)$,

$Er_x^{n+1}(\alpha\#x^m\$) = u^{n+1} x^{m-2(n+1)-2} u^{n+1}$.

And we prove this theorem by induction for all x^m.

$\circ$ Now, we have to prove it for $yx^m y$. This proof is not given because it is done on the same way as the previous one.

Therefore, the complete proof for each sequence on a word is on the same model.

$\diamond$

In this section, we introduced a rewriting system. We prove that our system stops for any words, and we show what kind of words it computes.

b. Dilatation

Now, let us define Di_x, another rewriting system on Σ^*. This system will *dilate* a sequence in a word when this sequence is surrounded with u letters.

Di_x : Dilatation with respect to x:

$$
\begin{array}{lcl@{\qquad\qquad}rcr}
\alpha\#x & \longrightarrow & \alpha x & \alpha\#y & \longrightarrow & \alpha y \\
\alpha\#u & \longrightarrow & \alpha u & \alpha xy & \longrightarrow & x\alpha y \\
\alpha xx & \longrightarrow & x\alpha x & \alpha xu & \longrightarrow & xx\alpha \\
\alpha yx & \longrightarrow & y\alpha x & \alpha yz & \longrightarrow & y\alpha z \\
\alpha yu & \longrightarrow & y\alpha u & \alpha ux & \longrightarrow & x\alpha x \\
\alpha uy & \longrightarrow & u\alpha y & \alpha uu & \longrightarrow & u\alpha u \\
\alpha x\$ & \longrightarrow & x\alpha\$ & \alpha y\$ & \longrightarrow & y\alpha\$ \\
\alpha u\$ & \longrightarrow & u\alpha\$ & \alpha\#\$ & \longrightarrow & \alpha\$ \\
\alpha\$ & \longrightarrow & \varepsilon & & &
\end{array}
$$

This rewriting system represents the dilatation of a word relatively to the letter x, with x as a particular element on $\mathcal{L}$ and $y, z \in \mathcal{L}$ $y \neq x, z \neq x$.

We present some theorems analogous to those previously exhibited for Er_x. The proof for each theorem is identical with the proof of the corresponding theorem of Er_x.

First, we present a theorem to prove the correctness of this rewriting system.

Theorem 6 *$\forall v \in \mathcal{L}_u^*, \forall x \in \mathcal{L}$, $Di_x(\alpha\#v\$)$ exists and is unique.*

Proof of theorem 6

The proof of this theorem is identical with the proof of theorem 4.

$\diamond$

Now, we want to determine which kind of modifications are computed by means of these rules.

Theorem 7 *Let v be a word in $\mathcal{L}_u^*$. Let $n > 0$, $0 \leq m \leq n$ and $0 \leq p \leq n$.*

A sequence $u^m x u^p$ in v becomes a sequence x^{m+p+1} in $Di_x^n(\alpha\#v\$)$.

Proof of theorem 7

The proof of this theorem is identical with the proof of theorem 5.

$\diamond$

Now, with these two rewriting systems we will define two usual operations from the mathematical morphological: the *opening* and the *closure*.

c. Opening

The opening is the composition $Di_x \circ Er_x$ of the two previous operators (i.e. rewriting systems). This composition allows to erase small sequences of letters in a word.

Definition 8 (opening) *Let $v \in \mathcal{L}_u^*$, $x \in \mathcal{L}$ and $n \in I\!N$.*

The n-opening of the word v with respect to x is defined as:

$$Op_x^n(v) = Di_x^n(\alpha\#\ Er_x^n(\alpha\#v\$)\ \$)$$

For example, with the word $v = hhlllhlhl$, we have $Op_h^1(v) = uulllulul$ and $Op_l^1(v) = hhlllhuhu$.

The principal interest of this operation for us lies in the following theorem:

Theorem 8 *Let $v \in \mathcal{L}_u^*$, $x \in \mathcal{L}$ and $n \in I\!N$. Let x^m, with $m \in I\!N$, be a sequence in v.*

If $m \leq 2n$ then x^m becomes a sequence u^m in $Op_x^n(v)$.

If $m > 2n$ then x^m remains unchanged in $Op_x^n(v)$.

Proof of theorem 8

$\circ$ The first part of this theorem is obvious.

From theorem 5, a sequence x^m with $m \leq 2n$ becomes a sequence u^m with the erosion. Then, from theorem 7, a sequence u^m stays unchanged after the dilatation.

Therefore, if $m \leq 2n$ then x^m becomes a sequence u^m.

$\circ$ It is similar to prove the second part.

From theorem 5, a sequence x^m with $m > n$ becomes a sequence $u^n x^{m-2n} u^n$ with the erosion. And, from theorem 7, a sequence $u^n x$ becomes a sequence x^{n+1} with the dilatation. Thus, $u^n x^{m-2n} u^n$ becomes a sequence $x^{n+m-2n+n} = x^m$.

Therefore, if $m > 2n$ then x^m remains unchanged.

And we prove the theorem.

$\diamond$

The n-opening of a word is a word in which all small sequences are erased. A sequence is small when its length is less than $2n$. The advantage of this operation is that we can erase all sequences in v with a length smaller than a fixed value.

d. Closure

Further, the closure is the composition $Er_x \circ Di_x$ of the two operators of erosion and dilatation. This composition allows to join sequences of letters in a word if these sequences are separated by less than $2n$ letters u.

Definition 9 (closure) *Let $v \in \mathcal{L}_u^*$, $x \in \mathcal{L}$ and $n \in I\!N$.*

The n-closure with respect to x of the word v is defined as:

$$Cl_x^n(v) = Er_x^n(\alpha\#\ Di_x^n(\alpha\#v\$)\ \$)$$

We have the following theorem to explain what we can obtain with this operator.

Theorem 9 *Let $v \in \mathcal{L}_u^*$, $x \in \mathcal{L}$ and $n \in I\!N$*

A sequence $u^n x u^m x u^n$, with $m \in I\!N$, becomes a sequence $u^n x^{m+2} u^n$ in $Cl_x^n(v)$ if $m \leq 2n$.

Proof of theorem 9

From theorem 7, the sequence $u^n x u^m x u^n$ becomes the sequence x^{n+m+2} after the dilatation.

Then, from theorem 5, the sequence x^{n+m+2} becomes the sequence $u^n x^{m+2} u^n$ after the erosion.

Therefore, we prove the theorem.

$\diamond$

Then, two sequences separated by less than $2n$ letters u are unified. In fact, the n-closure allows to transform the small sequences of u into a letter of $\mathcal{L}$.

Finally, we introduce our fundamental operator, the *filter*, which transforms a word into a sequence of homogenous series of letters of $\mathcal{L}_u$. In the framework of the utilization of a training set, a filter allows to smooth the training set to deduce a fuzzy partition.

e. Filter

A filter is a composition of the precedent operators to transform words.

Definition 10 (filter) *Let $v \in \mathcal{L}_u^*$, $x \in \mathcal{L}$ and $n \in I\!N$*

The n-filter with respect to x of the word v is defined as:

$$\begin{array}{ll} \text{if } n = 1 & Fil_x^1(v) = Cl_x^1(Op_x^1(v)) \\ \text{if } n > 1 & Fil_x^n(v) = Cl_x^n(Op_x^n(Fil_x^{n-1}(v))) \end{array}$$

The particular combination of these operators generates some interesting properties. A filter will allow to *smooth* a word. Firstly the sequences with a length of $2n$ letters are erased (*i.e.* replaced by the letter u), then we unify sequences separated by $2n$ letters.

We will show in the next section our fundamental algorithm for which all these operators have been defined.

3 Fuzzification of the set of values

In this Section, we present an algorithm to infer a fuzzy partition on a set of numerical values, after the use of the previous rewriting system.

When we apply a filter upon the word induced by a training set, we are able to eradicate small sequences of classes. Then, if we apply progressively this filter, from 1 to n, we will be able to smooth a long word.

Algorithm 2 *To smooth a training set*
If $v \in \mathcal{L}_u^*$ is the word induced by a training set, $\forall x \in \mathcal{L}$, apply $Fil_x^n(v)$.

After the use of this algorithm, we have a word with large sequences (with a length larger than $2n$). n is a fixed value given to the system. For example, we use $n = 10$ percent of the size of the training set.

The sequences of u represent *uncertain* sequences where the classes are highly mixed. The sequences of $x, x \in \mathcal{L}$, are sequences with a single class, in a certain point of view, these sequences contain no u character. We call them *certain* sequences, whatever x may be. We will use these two kinds of sequences to build a fuzzy partition related to an attribute, on the training set $\mathcal{S}$.

Let r be the number of fuzzy modalities we want for the attribute. We select the r largest sequences containing one class, for instance $[\mathcal{S}_1^{min}, \mathcal{S}_1^{max}]$ and $[\mathcal{S}_2^{min}, \mathcal{S}_2^{max}]$ when $r = 2$. In the case where we cannot find r such sequences, we can either reduce the number of applied filters, or select less sequences. We infer a fuzzy partition upon the universe of data with these r sequences: the considered intervals define the kernel of each fuzzy modality (*Fig. 4*). We summarize this in the following algorithm, with $r = 2$:

Algorithm 3 *To fuzzify a training set with regard to an attribute in 2 fuzzy subsets*

1. *Translate $\mathcal{S}$ into a word v with algorithm 1.*
2. *For n fixed, smooth v with algorithm 2.*
3. *Find the two larger certain sequences $\mathcal{S}_1$ and $\mathcal{S}_2$.*
4. *Let $\mathcal{S}_1^{min}$ and $\mathcal{S}_1^{max}$ (resp. $\mathcal{S}_2^{min}$ and $\mathcal{S}_2^{max}$) be the values which define $\mathcal{S}_1$ (resp.$\mathcal{S}_2$). $\mathcal{S}_1 \equiv [\mathcal{S}_1^{min}, \mathcal{S}_1^{max}] (resp. \mathcal{S}_2 \equiv [\mathcal{S}_2^{min}, \mathcal{S}_2^{max}])$ with $\mathcal{S}_1^{max} < \mathcal{S}_2^{min}$. We denote by $\mathcal{S}_i^{min}$ (resp. $\mathcal{S}_i^{max}$) the value associated with the first (resp. last) letter of $\mathcal{S}_i$ in X_j^d.*
5. *the fuzzy partition is defined as a family of two fuzzy subsets. The kernel of the first one is $]-\infty, \mathcal{S}_1^{max}]$ and its support is $]-\infty, \mathcal{S}_2^{min}]$. The kernel of the second one is $[\mathcal{S}_1^{max}, +\infty[$ and its support is $[\mathcal{S}_2^{min}, +\infty[$*

Thus, we have an algorithm to induce a fuzzy partition from a training set. This algorithm smoothes a word induced by a training set and, from this smoothed word, we propose a way to define a fuzzy partition.

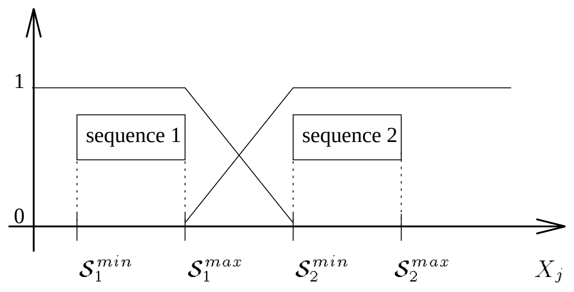

Figure 4: Induced fuzzy partition

Conclusion

In this paper, we propose a new algorithm to infer a fuzzy partition upon the universe of a set of numerical values. This algorithm leads on several rewriting systems. Each of these rewriting systems is modeled on a mathematical morphology operator. We have a system to reduce any sequence of letters and a system to enlarge any sequence of letters. When we compose these two systems we obtain two general operators depending on the order of the composition. These two operators allow us to control the way of erasure of small mixed sequences. Finally, we obtain an algorithm to smooth a word induced by a training set, and, from this word we propose a way to define a fuzzy partition.

This method is implemented in our decision trees builder program to help the construction of decision trees when no knowledge from expert is available. We use this partitioning as an additional step of the Ramdani's algorithm SAFI.

Some future features can be to take into account another dimension of the set of values, which is the distance between two consecutive values, and to adapt the fuzzy mathematical morphology to this kind of problems.

References

[Autebert87] Autebert (J. M.). – *Langages Algébriques.* – Paris, Masson, 1987.

[BM et al.95] Bouchon-Meunier (B.), Marsala (C.) et Ramdani (M.). – Arbres de décision et théorie des sous-ensembles flous. *In : Actes des 5èmes journées du PRC-GDR d'Intelligence Artificielle*, pp. 50–53.

[Breiman et al.84] Breiman (L.), Friedman (J. H.), Olshen (R. A.) et Stone (C. J.). – *Classification And Regression Trees.* – New York, Chapman and Hall, 1984.

[Carter et al.87] Carter (C.) et Catlett (J.). – Assessing credit card applications using machine learning. *IEEE Expert*, vol. Fall Issues, 1987.

[Catlett91] Catlett (J.). – On Changing Continuous Attributes into Ordered Discrete attributes. *Lecture notes in Artificial Intelligence*, vol. 482, 1991, pp. 164–178.

[Coster et al.89] Coster (M.) et Chermant (J. L.). – *Précis d'analyse d'images.* – Presses du CNRS, 1989.

[Fayyad et al.93] Fayyad (U. M.) et Irani (K. B.). – Multi-interval discretization of continuous-valued attributes for classification learning. *In : Proceedings of the 13th I.J.C.A.I.*, pp. 1022–1027.

[Ginsburg66] Ginsburg (S.). – *The Mathematical Theory of Context Free Languages.* – New-York, McGraw-Hill, 1966.

[Jang94] Jang (J-S. R.). – Structure determination in fuzzy modeling: a fuzzy CART approach. *In : Proceedings of the 3rd IEEE Int. Conf. on Fuzzy Systems*, pp. 480–485.

[Janikow94] Janikow (C. Z.). – Fuzzy decision trees: FIDMV. *In : Proceedings of the Joint Conf. on Information Science.*

[Maher et al.93] Maher (P. E.) et Saint-Clair (D.). – Uncertain reasonning in an ID3 machine learning framework. *In : 2nd IEEE Int. Conf. on Fuzzy Systems*, pp. 7–12.

[Marsala94] Marsala (C.). – *Arbres de décision et sous-ensembles flous.* – Rapport 94/21, LAFORIA-IBP, 1994.

[Quinlan86] Quinlan (J. R.). – Induction of decision trees. *Machine Learning*, vol. 1 (1), 1986, pp. 86–106.

[Quinlan93] Quinlan (J. R.). – *C4.5: Programs for Machine Learning.* – San Mateo, Ca, Morgan Kaufmann, 1993.

[Ramdani92] Ramdani (M.). – Une approche floue pour traiter les valeurs numériques en apprentissage. *In : Journées Francophones d'apprentissage et d'explication des connaissances.*

[Ramdani94] Ramdani (M.). – *Système d'Induction Formelle à Base de Connaissances Imprécises.* – Thèse de l'Université P. et M. Curie, rapport 94/1, LAFORIA-IBP, 1994.

[Serra82] Serra (J.). – *Image Analysis and Mathematical Morphology.* – New York, Academic Press, 1982.

[Shannon et al.49] Shannon (C. E.) et Weaver (W.). – *The Mathematical Theory of Communication.* – Urbana, University of Illinois Press, 1949.