Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection, CNRS, UMR 7005

Learning Classifier Systems for Hyperspectral Image Processing

A. Quirin, J. Korczak, M. V. Butz, D. E. Goldberg

Rapport de recherche ULP-LSIIT-RR-2004-01

Illkirch, Mai 2004

Learning Classifier Systems for Hyperspectral Image Processing

A. Quirin, J. Korczak, M. V. Butz, D. E. Goldberg

A. Quirin, J. Korczak
LSIIT, CNRS/Université Louis Pasteur,
Pôle API, Boulevard Sébastien Brant
F-67400 Illkirch cedex
{korczak,quirin}@lsiit.u-strasbg.fr

M. V. Butz, D. E. Goldberg
Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL, 61801
{butz,deg}@illigal.ge.uiuc.edu

Abstract. In this article, two learning classifier system based classification techniques are described to classify remote sensing images. Usually, these images contain voluminous, complex, and sometimes erroneous and noisy data. The first approach implements ICU, an evolutionary rule discovery system, generating simple and robust rules. The second approach applies the real-valued accuracy-based classification system XCSR. The two algorithms are detailed and validated on hyperspectral data. The comparison of the system includes differences in evolution accuracy and parameter refinement.

Keywords : Remote sensing image, classification rules, high resolution image, hyperspectral image, supervised learning, evolutionary learning, genetic algorithm, classifiers system.

Résumé. Dans cet article, deux méthodes de classification d'images de télédétection, basées sur les systèmes de classifieurs LCS, sont décrites. Généralement, ces images contiennent des données volumineuses, complexes et parfois erronées et bruitées. La première approche met en œuvre ICU, un système de découverte de règles évolutionnaire, produisant des règles simples et robustes. La seconde approche applique le système de classifieurs XCSR, basé sur des vecteurs à valeurs réelles. Les deux algorithmes sont détaillés et validés sur des données hyperspectrales. Enfin, ils sont comparés en termes de qualité d'évolution et de raffinement de leur paramètres.

Mots-clés : Images de télédétection, règles de classification, image haute résolution, image hyperspectrale, apprentissage supervisé, apprentissage évolutif, algorithme génétique, système de classifieurs.

1 Introduction

The emergence and the improvement of remote sensing, aircraft simulation, airborne and spaceborne sensor systems as well as other kinds of such survey technologies has considerably enhanced our means to explore and to collect data. However, this rapid increase in data results in more time and cost for storage as well as for the analysis and the mining of the data. At the same time, a lot of useless information can hide valuable information. These observations force data miners to focus on elaborated and sophisticated algorithms to overcome this rapid data growth.

For many years, the design of efficient and robust image classification algorithms has been the most important issue addressed by remote sensing image users. Strong effort has been devoted to elaborate new classification algorithms and improve techniques used to classify statistical data sets [Bock, 1999; Lanzi, 1997; Wilson 1995; Wilson 1998]. Relatively few data miners in the machine learning community have considered how classification rules might be discovered from raw and expertly classified images. This paper presents the potential contribution of evolutionary-based techniques to discover such rules. The unique source of information is a remote sensing image and its corresponding classification furnished by an expert. The images, registered by various satellites (e.g. SPOT, CASI, Quick Bird), generally contain voluminous data. Sometimes they are very noisy due to the presence of various details in a high spatial resolution or unfavorable atmospheric conditions at the time the images were acquired. These data can embrace different cameras having various spectral and spatial resolutions [Quirin 2002; Korczak 2003a; Weber, 1995; DAIS 2001].

The aim of this research is the performance evaluation of others classifier systems in a domain of remote sensing. It appears that learning classifier systems are well suited to remote image mining. They generate classification rules able to adapt themselves according to the available data, environment, and the evolution of classes. This research reports on two evolutionary classifier systems: ICU and XCS. ICU, developed at the LSIIT [Quirin, 2002], is an evolutionary rule discovery system combining a genetic algorithm and a population of classification rules describing constraints for each pixel from the data. XCS is a learning classifier system, developed by [Wilson, 1995; Wilson, 1998], that evolves a rule set online based on prediction accuracy and a niched genetic reproduction [Lanzi, 1997]. The two algorithms were tested on hyperspectral remote sensing data.

The paper is structured as follows. Basic terms and properties of hyperspectral images are introduced in Section 2. Section 3 describes the two algorithms ICU and XCS. In this section, the main components and the quality measures of the two methods are explained. A CASI image is analyzed in section 4 and comparisons are presented in section 5.

2 Hyperspectral remote sensing images

An hyperspectral image is a set of two dimensional arrays $I_{X,Y,S}$ where *(X,Y)* is respectively the width and the height of the image and S the number of spectral channels (or spectral bands). The term *hyperspectral* refers to an image which includes more than 20 spectral bands (similar to those produced by ROSIS and DAIS [DAIS01]). Conversely, the term *multispectral* is used in the case of a low number of spectral bands, as CASI and Quick Bird remote sensors [Toutin, 2002]. A value $I(x, y, s)$ in this array is the reflectance observed on the pixel location *(x,y)* at the wavelength corresponding to the spectral channel *s*. A reflectance value corresponds to the intensity of the response obtained from the ground. The input space of a classification problem can be viewed as an ordered vector of real numbers. For each pixel, the spectral signature of this pixel was used. Fig 1 shows the spectrum of reflectance of a pixel from a hyperspectral sensor (type MIVIS). Each spectral channel has roughly 10 nanometers in width.

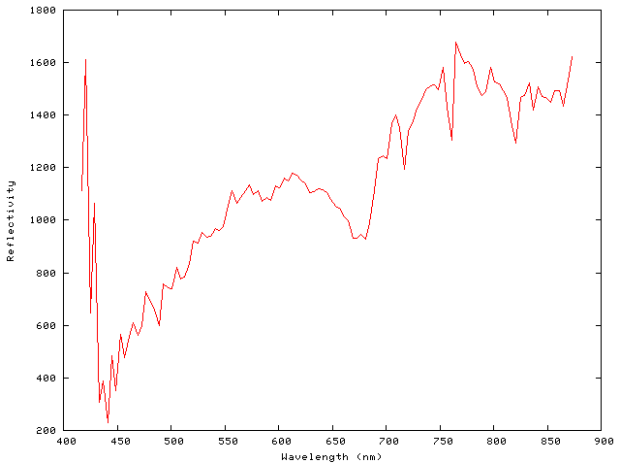

Fig 1 : Spectrum of reflectance observed for a pixel from the hyperspectral sensor MIVIS

The image data is very voluminous; typically 20 to 200 spectral channels in an image and their size can reach 8000 x 12000 pixels. Sometimes, half of bands is noisy because of sensor defects or atmospheric absorption of reflectance value in low wavelengths (see Fig 2). It can be noticed that the analysis of this data provides a combinatorial problem.

To illustrate our approach, two kinds of multispectral images have been used:

- *CASI data*. Airborne spectrometer, 1175x673, 15 spectral channels (0.43µm - 0.87µm), high resolution (1.3m). This image has been pre-processed by geometric correction and warping by first order polynomial warping and nearest neighbor re-sampling.

– *Quick Bird data*. Satellite, 619x238, 4 spectral channels (0.45µm - 0.90µm) in multispectral or one channel in panchromatic, very high resolution (2.8m in multispectral, 0.61m in panchromatic). Same pre-processing has been applied as before. The advantage of this sensor resides in the ability to be able to reprogram the analyzed wavelengths.

Learning and testing were applied on subsets of these images. Subsets contain 142x99 points, and only 1540 points were validated by human ground truthing (expertise ratio: 11%). Then, according to the validation strategies used (hold-one-out, cross-validation), testing sets represent 20% to 50% of the original validated image points. Fig 3 shows Quick Bird data for the Lagoon of Venice and the corresponding set of validated points. The orange rectangle is the area in which all validated points are situated. It should be noticed that the proportion of these points to the whole image is very small - about 0.01%.

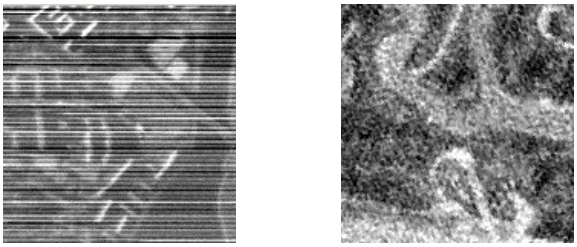

Fig 2 : Typical noisy channels (DAIS, Strasbourg and Lagoon of Venice)

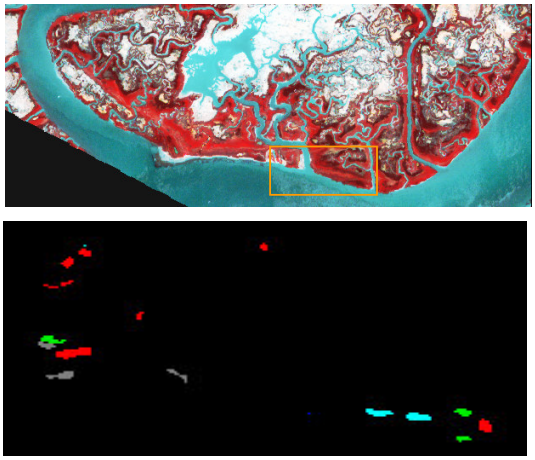

Fig 3 : Quick Bird data of the Lagoon of Venice and the corresponding set of validated points (ground truthing)

3 Evolutionary methods to discover the classification rules

3.1 Idea of rule discovery

The idea of rule mining is to discover a pool of classifiers, where each classifier, taking a pixel as input, returns its class. The discovery of a classification rule is a combinatorial problem. The search space is very large,.it depends on the size of the image, the pixel spectral resolution, and the rule encoding. Evolution-based approaches and particularly Learning Classifier Systems (LCS, [Horn, 1994]) are specifically designed to search efficiently the space of classifiers.

For a long time, many generic classifiers systems were developed to explore the potential of these techniques including ZCS [Wilson, 1994], LFCS [Bonarini, 2000], S-classifiers [Lanzi, 1999], ICU [Quirin, 2002] and XCS [Wilson, 1995; Wilson, 1998]. Two systems have been arbitrary chosen, ICU and XCS. ICU is an evolutionary rule discovery system combining a genetic algorithm, a population of classification rules

describing constraints for each pixel from the data, and a set of methods used in rules selection and accuracy assessment. In XCS, fitness is based on prediction accuracy and the classical binary strings of classifier systems were replaced by real-values vectors [Wilson, 2000]. In the following sections, the main components of ICU and XCS are described.

3.2 ICU

3.2.1 Overview of the algorithm

To discover a classification rule, ICU uses a Michigan-like learning classifier system representation, in which each rule is encoded by one individual [Quirin, 2002]. Moreover, separate pools are used for each class to discover classification rules. Only one rule is kept for each class at the end of a run. Steady-state strategy is used for reproduction and replacement. The pool contains only 100 to 200 individuals for the discovery of each class and 2000 generations were used. The most common values for the crossover and mutation rate were tested (e.g. $(\chi;m) = (0{,}8;0{,}05)$). After having tested a number of selection methods as randomly, elitism, roulette wheel and tournament, a roulette wheel selection based on the rank in the pool of the individual's fitness was retained for the replication of individuals and tournament for generational replacement.

Classification rules are symbolic expressions and describe conditions to be held and actions to be taken if the conditions are satisfied. From a functional point of view, a rule represents a piece of knowledge about a class by a conditional expression, such as *if <conditions> then <class>*. The "*condition*" part of a rule specifies a constraint in the system such as value, color, form, shape, etc, corresponding to conditions that must be fulfilled in order to activate the rule. The "*class*" part defines the class of the instance currently treated by the rule given the appropriate conditions are satisfied. One asserts that the evolved rules should be rapidly evaluated and easy to interpret by any user. As a result, condition representation using the concept of an interval could be fully adequate for remote sensing image classification. In terms of machine learning, the rules have to be absolutely specific, meaning that they have to cover the extreme maximum and minimum pixels belonging to any given class. A short discussion about the representation of the rules follows, but we refer the reader to [Korczak, 2003] for more information.

Before rule specification, recall that a pixel is encoded as a spectral vector, describing values of reflectance for the *n* bands of the remote sensing image, i.e. a pixel can be considered as a point in a R^n space :

$$< pixel > := [b_1\ b_2\ b_3 \ldots b_n] \qquad [1]$$

In our system, the condition of any rule is built on the concept of spectral intervals defining a given band corresponding to a given class. Such intervals are a pair of integer numbers, between 0 and the maximum possible value for a pixel of a given band (i.e. 65536 for pixels defined on 16 bits). This solution allows partitioning the space of the spectral values in two ranges: the first containing the pixel values corresponding to a given class, and the second containing the remainder.
To precisely specify the class definition, a set of intervals is defined for each band of the remote sensing image. Taking into consideration all bands, the condition part is defined as a set of hyper-rectangles in a R^n space :

$$< condition > := \bigwedge_{i=1}^{n} \bigvee_{j=1}^{k_i} (\mathrm{m}_i^j \le b_i \le \mathrm{M}_i^j) \qquad [2]$$

where m_i^j and M_i^j denote, respectively, the minimal and maximum reflectance values allowed for a pixel belonging to a class *C* for band *i*. *k* is a fixed parameter which defines the maximum number of disjunctions allowed.

The chosen representation is mainly due to simplicity, compactness and uniform encoding of spectral constraints. During experimentation, the representation has demonstrated rapid execution of genetic operators and efficient computing. Of course, one may specify more complex structures using contextual or temporal features, however they not only require more sophisticated genetic operators but also more powerful computers to perform the calculation in an acceptable amount of time. Additionally, the more complicated the representation structure, the more the system might over-fit the data poorly generalizing over new instances.

3.2.2 *Genetic operators*

Rule initialization

Generally, in remote sensing, the initial population of classification rules is randomly created from raw images and given classes, and then evolved by a genetic algorithm. In this study, in order to efficiently develop the classification rules, a genetic algorithm initializes interval values according to spectral limits of the classes designated by an expert for valid zones of the remote sensing image. More about initialization algorithms can be found in (Kallel, Schoenauer, 1997). Rule initialization considers the values of reflectance most frequently recorded in the spectra of a given class. Fig 4(a) presents spectrograms for 5 classes (shown in different colors). Quick-Bird data (4 spectral channels, each one in his column) were used. The figure shows the most frequent values for a given wavelength in black.

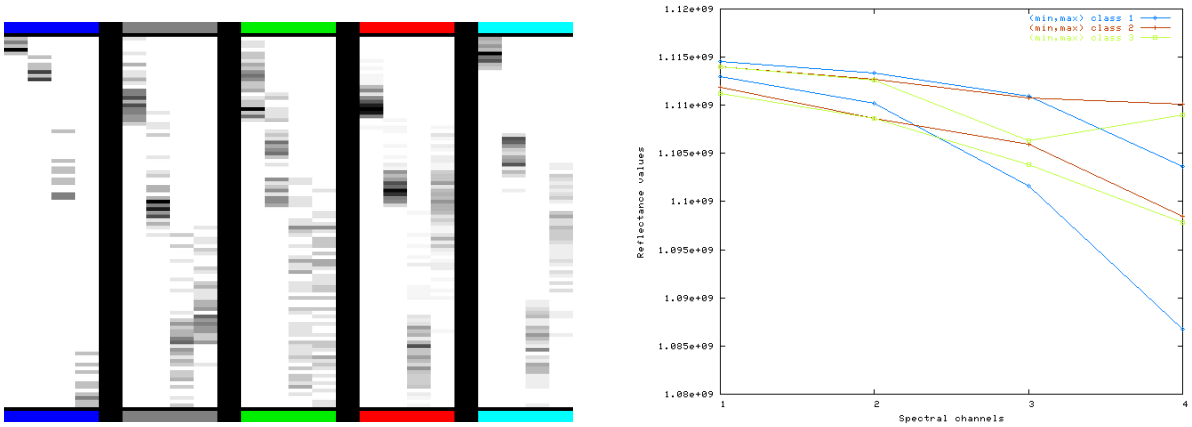

Fig 4 (a) Spectrogram representation of the data and (b) First generated rules for 3 choosen class

Initial classification rules are created based on the maximum and minimum values observed on the spectrogram for each class. Fig 4(b) represents the first generated rules for three chosen class (light blue, dark blue and gray), corresponding to the spectrograms of Fig 4(a). Two algorithms for the generation of a pool of rules have been proposed, according to the expected solution given by the expert, but allowing also some diversity. The first, called MinMax, creates maximum intervals covering all the pixels belonging to a given class, and the second algorithm, called Spectro, integrates the spectral distribution density and interval partitioning. The proposed rule initialization seems to be well suited for large volumes of data considerably reducing the search space by generating initial rules close to the final solution.

Crossover operator

Crossover requires two rules, and cuts their chromosome at some randomly chosen positions to produce two offspring. In fact, crossover exchanges hyper-rectangles at a randomly selected *level* in these rules. A *level* is the depth of conjunction or disjunction in a rule, as if it was represented by a hierarchical tree, as in Genetic Programming. The two new rules inherit some rule conditions from each parent rules. A crossover operator is used in order to exploit the knowledge of the rules. Each result of the crossover process has to be validated. Validation of the various rule attributes (border limits violation, overpassing, etc) is carried out by a process of interval merging. However, merging not only decreases the number of intervals in the rules, but also generates some information loss. In fact, in order to avoid premature convergence of rules, it is generally important to preserve two distinct intervals instead of a single aggregated one for the following generation. On the other hand, it is interesting to note that the positive or negative effects of an interval on the quality of the rule can be related to other intervals encoded in the classification rule.

Mutation operator

The mutation operator plays a dual role in the system: (1) it provides and maintains diversity in the population, and (2) in combination with selection, it performs local searches. The operator is a bit more complex than the previous one. Mutation randomly selects one operation in a *library of operators* and applies it to a rule selected with a given selection scheme. The selected operator can be: (1) suppression of a

spectral channel (band mutation), (2) addition or suppression of a constraint, and (3) shift or cut of a constraint in the spectral definition space of the data (interval mutation). The complexity of this operator including all the sub-operators ensures that a more diverse offspring population will be obtained.

Band mutation consists of a deletion of spectral bandwidth. Its interest is twofold; first, it enables simplification and generalization of a rule; secondly, it allows the elimination of noisy bands that frequently appear in hyper spectral images. The fact, there is no addition of a band (algorithm must add several constraints in many times) ensuring that generalization is the preferred behavior of the evolutionary process.

Interval mutation allows the addition of a chosen band, eliminating or cutting an interval in two spectral ranges. In case of addition, the new rules are completed by a new intervals centered randomly with a user-defined width. The cutting of an interval is done by random selection of a cutting point within the interval (for example, the cutting of [10; 100] can generate two intervals: [10;15] and [15;100]). Mutation such as this allows for breakage of continuous spectral ranges, observed in true data.

Stop criterion

The evolution process converges according to some statistical criteria indicating if the current rule is near to a global optimum or if the population of rules will not evolve anymore. In our system, not only the evolution of quality of the best discovered rule is taken into consideration, but also a minimum acceptable quality defined by a user, a process stability measure, and a maximal number of generations. If one of these criteria is satisfied, then the process is stopped. One of the stop criteria used is based on stabilization detection (see Eq. [3])

$$\left| \frac{\sum_{k=1}^{P} Q_k}{P} - Q_0 \right| \leq E \qquad [3]$$

where Q_0 is the fitness of the best rule in the current generation and Q_k is the quality of the best rule obtained during the last k generation. Q_0 is compared to the mean of Q_k for P generations ($1 \leq k \leq P$). The genetic algorithm is stopped when the difference falls below a threshold. P represents the maximum period of quality stabilization and E is a maximal variation of this stabilization compared with the current quality. The measure effectively detects the stabilization of fitness.

3.2.3 Qualities Measures

In this section, the principal quality measures, which are commonly used to evaluate the quality of classifiers, are introduced. The measures strongly depend on the application domain. In image classification, the evaluation is usually based on the confusion matrix containing the classification produced by the classifier (I_{class}) and the classification given by an expert (I_{expert}). Table 1 defines the variables necessary to compute this measure.

Table 1 : Confusion matrix for computing the quality measures

		Expert classification or actual values (I_{expert})					Q_{ppa}	Number of pixels (k_i)
		C_1	C_2	⋯	⋯	C_n		
Image classified by the classifier (I_{rule})	C_1	$P_{C_1}^{C_1}$	$P_{C_1}^{C_2}$	⋯	⋯	$P_{C_1}^{C_n}$	Q_{ppa}^{1}	$\Omega(C_1)$
	C_2	$P_{C_2}^{C_1}$	$P_{C_2}^{C_2}$				⋮	⋮
	⋮	⋮		⋱			⋮	⋮
	⋮	⋮			⋱		⋮	⋮
	C_n	$P_{C_5}^{C_1}$				$P_{C_n}^{C_n}$	Q_{ppa}^{n}	$\Omega(C_n)$
Q_{sens}		Q_{sens}^{1}	⋯	⋯	⋯	Q_{sens}^{n}		

A very popular measure is a classifier accuracy that measures the proportion of correctly classified pixels with respect to all pixels for *n* classes;

$$Q_{accur} = \frac{\sum_{i=1}^{n} P_{C_i}^{C_i}}{\sum_{i=1}^{n} \Omega(C_i)} \quad [4]$$

The weakness of this measure is that it only takes correctly classified pixels into consideration. To make the classification results more specific, three other related measures are defined: Q_{ppa} positive predictive accuracy, Q_{sens} sensitivity and Q_{spe} specificity.

The positive predictive accuracy, Q_{ppa}, measures the classifier reliability of correctly classifying pixels compared to all pixels associated by a classifier to a given class:

$$Q_{ppa}^{i} = \frac{P_{C_i}^{C_i}}{\sum_{k=1}^{n} P_{C_i}^{C_k}} \quad [5]$$

The sensitivity (Q_{sens}) measures the fraction of correctly classified pixels with respect to all pixels classified by an expert to a given class (i.e. proportion of true positives).

$$Q_{sens}^{i} = \frac{P_{C_i}^{C_i}}{\sum_{k=1}^{n} P_{C_k}^{C_i}} \quad [6]$$

The specificity (Q_{spe}) measures the rate of correctly classified pixels not being in the given class (i.e. proportion of true negatives).

$$Q_{spe}^{i} = \frac{\sum_{j=1, j\neq i}^{n} \sum_{k=1, k\neq i}^{n} P_{C_k}^{C_j}}{\sum_{j=1, j\neq i}^{n} \sum_{k=1}^{n} P_{C_k}^{C_j}} \quad [7]$$

The four above defined measures may give a faithless image of classifier performance in case of non equal distribution of pixels over a set of classes. Therefore, it is recommended to use an additional measure of weighted accuracy computed as follows:

$$N_{final}^{i} = \alpha \cdot Q_{sens}^{i} + (1-\alpha) \cdot Q_{spe}^{i} \quad [8]$$

Parameter α allows the adjustment of the relative weight given to true positives and true negatives. As mentioned before, the measure is used for certain classes that are under- or over-represented. By default the value of this parameter equals to 0.5, which means that the same importance is given to both measures.
The proposed measure has a number of advantages; mainly it is independent of the pixel processing sequence, invariant of the size of classes, and effective for class discovery with a highly variable number of pixels.

The global quality measure is a weighted average of the N_{final}^{i} by the size of each class.

$$N_{global} = \frac{\sum_{i=1}^{n} \Omega(C_i) \cdot N_{final}^{i}}{\sum_{i=1}^{n} \Omega(C_i)} \quad [9]$$

All these quality measures have been tested for different values of α and were applied on confusion matrices or DCM (see section 5.1 for more information). The statistics in section 6 will be presented using a fixed value of α and the two kinds of confusion matrices.

3.2.4 Rule selection

A pixel passed to the system may activate several rules. The rules are evolved in separate pools and nothing is done until the end to force the learning of a given pool to acquire and block itself on a ground not cleared by the other classifiers. In this point of view, the other classifiers are not constrained by a strong but false rule who *would rob* their pixels. But at the end or a run, a selection mechanism is needed to choose the appropriate rule when more than one rule is available for a given pixel. Two methods have been developed that can be used also if no rules are activated by the pixel. Note that this operator acts only in the case when 0 or more than one rule is activated. If only one rule is activated, no modification of the result is done and the rule is kept as is.

The first one, called *BestScore*, simply selects the rule depending on the fitness value obtained from the last pass in the learning set. Among the *n* activated rules the rule with the best fitness is selected. If *n=0*, the number of constraints n_c of the rules (i.e. for each spectral channel) accepted by the pixel is tested, and the rule with the highest n_c wins. In case of equality, the first rule is chosen. This seems to be a naive method. The problem comes from the use of the fitness value. Since the value is not directly connected to the data, a skew may be introduced in the evaluation method. In fact, the method has shown good results with expertise data coming from unsupervised learning. In this case, the algorithm should deal with very accurate classes but also noisy classes (*waste* classes). In this case, selection by the means of fitness is very useful.

To avoid the use of values dependent only on the paradigm, a second method has been designed, called *CloseCenter*. The deviation of the spectrum of the pixel compared to the mathematical center of each constraint in the rule is computed. For *n>1*, the smallest deviation selects the gaining rule and for *n=0*, the same technique as before is applied. Note that *BestScore* and *CloseCenter* adapt dynamically to the data (for each pixel), but as fitness and spectrum represent a continuum between two close pixels, the obtained classification is in most cases coherent.

3.3 XCS

XCS evolves a set of rules, the so-called *population* of *classifiers*. Rules are evolved by the means of a GA. A classifier usually consists of a condition and an action part. The condition part specifies when the classifier is applicable and the action part specifies which action, or classification, to execute. In contrast to the original LCSs, the fitness in the XCS classifier system, introduced by Wilson [Wilson, 1995], is based on the *accuracy* of reward predictions rather than on the reward predictions themselves. Thus, XCS is meant to evolve not only a representation of an optimal behavioral strategy, or classification, but rather to evolve a representation of a complete payoff map of the problem. That is, XCS is designed to evolve a representation of the expected payoff in each possible situation-action combination.

Since our system is confronted with real-valued data in this study, we apply the real-valued extension of XCS (XCSR) introduced in [Wilson, 2000]. Recently, several studies were reported that show that XCS performs comparably well to several other typical classification algorithms in many standard datamining problems [Bernadò, 2002; Dixon, 2002; Bernadò, 2003].

This section provides a short introduction to the XCS classifier system. For a more detailed introduction to XCS and XCSR the interested reader is referred to the original paper [Wilson, 1995], the real-valued extension [Wilson, 2000] and the algorithmic description [Butz, 2002].

3.3.1 Overview of the algorithm

As mentioned, XCS evolves a population [P] of rules, or classifiers. Each classifier in XCS consists of five main components:

1. The condition part specifies the subspace of the input space in which the classifier is applicable, or *matches*. In our real valued problem, a condition specifies a conjunction of intervals, one for each attribute. If the current problem instance lies within all specified intervals, the classifier matches.
2. The action part specifies the advocated action, or classification.
3. The payoff prediction estimates the average pay-off encountered after executing action A in the situations in which the condition part matches.

4. The prediction error estimates the average deviation, or error, of the payoff prediction.
5. The fitness reflects the scaled average relative accuracy of the classifier with respect to other overlapping classifiers.

Learning usually starts with an empty population. Given current input, the set of all classifiers in [P] whose conditions match the input is called the match set [M]. If some action is not represented in [M], a covering mechanism is applied. Covering creates classifiers that match the current input and specify the not covered actions. Given a match set, XCS can estimate the payoff for each possible action forming a prediction array P(A). Essentially, P(a) reflects the fitness-weighted average of all reward prediction estimates of the classifiers in [M] that advocate classification a. The payoff predictions determine the appropriate classification. During learning, XCS chooses actions randomly. During testing, the action a_{max} with the highest value $P(a_{max})$ is chosen.

3.3.2 Reinforcement and discovery components

XCS iteratively updates its population of classifiers with respect to the successive problem instances. After the classification is selected by the means of the prediction array and applied to the problem, scalar feedback is received. In a classification problem, classifier parameters are updated with respect to the immediate feedback in the current action set [A], which comprises all classifiers in [M] that advocate the chosen classification a. After rule evaluation and possible GA invocation, the next iteration starts.

The aforementioned covering mechanism ensures that all actions in a particular problem instance are represented by at least one classifier. Each attribute of the new classifier condition is initialized using parameter cover-rand that specifies the maximal interval the condition comprises in an attribute. XCS applies a GA for rule evolution. A GA is invoked if the average time since the last GA application upon the classifiers in [A] exceeds a threshold. The GA selects two parental classifiers using set-size relative tournament selection [Butz, 2003]. Two offspring are generated reproducing the parents and applying crossover (uniform crossover) and mutation. Parents stay in the population competing with their offspring. In the insertion process, subsumption deletion may be applied [Wilson, 1998] to stress generalization. Due to the possible strong effects of action-set subsumption, we only apply GA subsumption, which searches for an accurate, more general classifier that may subsume the offspring. If such a more general classifier is found, the offspring is discarded and the numerosity of the subsumer is increased. The population of classifiers [P] is of fixed size N. Excess classifiers are deleted from [P] with probability proportional to an estimate of the size of the action sets that the classifiers occur in. If the classifier is sufficiently experienced and its fitness F is significantly lower than the average fitness of classifiers in [P], its deletion probability is further increased.

3.3.3 Rules selection

As in ICU, the same pixel may activate several rules coding for different classes. This behavior can be constrained for solving problems as *one-to-one classification*. Two methods (*MaxConfident* and *ScoredConfident*) were tested, asking the pool to give a unique class for each pixel:

1. In the first one, only the rules which have a correct self-confidence are considered, in other terms, payoff prediction should be greater than a given threshold (fixed in our experiments to the half of the maximum value of payoff prediction for the current problem). Then the most frequent class obtained from rules which had matched the pixel is returned.
2. In the second one, all the rules which have matched the pixel are considered. For a class *c*, a score is computed as follows for each rule *r*:

$$s_r = \frac{\sum (P_r * F_r)}{\sum F_r} \qquad [10]$$

where P_r is the prediction payoff of the rule *r* and F_r its fitness. The action of the rule with the best S_r is returned.

These two methods were experimented for different subsets of CASI data and for a pre-treated expertise set as follows: if more than one class was affected to the same pixel, only the dominant class was retained according to the percentage of its concentration given by the expert.

4 Case studies

4.1 Consideration with the data

The data available for this study was acquired during a field campaign at the end of September 2002, for the European project TIDE (Tidal Inlets Dynamics and Environment, [TIDE, 2001]). During this project, data was obtained from satellite or aircraft, at different scales and resolutions, providing a *multi-level view* of the ground [Stow, 2004]. A multi-level remote sensing is a useful technique to monitor large areas: a global view of the area can be identified by the satellite image, while the checking of a particular area or a classification need aerial imagery. The expertise used in the following supervised classification are based on a costly mean, ground truthing (the characterization of all points are made by hand by a human expert), and on expert validation provided by the examination of different levels of the data. Due to the existence of such different sources, the expert validation of points is not the easiest way but it is relatively safe and relevant. Some assumptions were made about the data [Dave, 2002]:

- the reference data in the learning base are truly representative of the sought classes;
- the reference data and the expert data are perfectly synchronized by geo-referencing;
- there is no error in the reference data (incorrect or missing class assignments, change in the vegetation boundaries between the time of imaging and the time of field verification, positional errors, ...).

Considering the expertise, two issues have to be addressed. (1) The complexity of the analyzed environment, particularly the effect of partial volume (data includes *pixels* of non pure classes), requires that the expert selects several classes for one point (*multi-labeling*). A typical supervised method handles only one class attribute. Thus, in the case of multi-labeling, the dominant class is kept or the point is dropped. Section 5.1 is a short discussion about the benefits brought by ICU and XCS for this point. (2) Mainly for cost reasons, the human expert can only label very few points. The identification of the boundary of each class and the construction of convex hulls are mandatory to include more and interior pixels in the expertise. The points can represent exactly the class, a part of the class, or sometimes a corner of a different kind of vegetation cross a hull supposed to be a whole class. Extra knowledge may be needed to select the correct points.

In this paper, a remote sensing image of San Felice (Lagoon of Venice) has been chosen (Fig 5). This image contains multispectral data (CASI 15 bands), with 142x99 pixels, 16 bits per pixel and 1.3m terrain resolution (Quirin, 2002). The case study considers a typical problem of classification for rural zones, with only five requested classes but including a high percentage of mixed pixels. Learning was carried out on 50% of the 1540 points, and then validation was performed on the whole data.

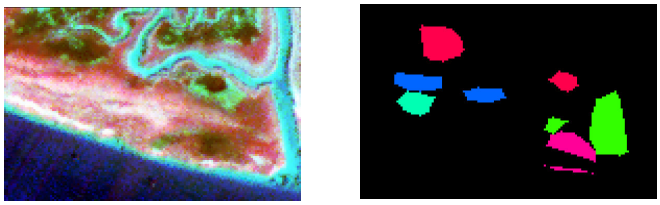

Fig 5: Reference and expert data (CASI, San Felice)

4.2 ICU classification

The following image and statistics resume our experiments.

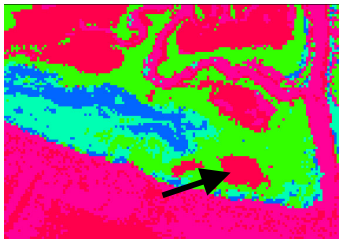

Fig 6: Classified image

Table 2: Confusion matrix

		Classification by the expert					
		SAR2	LISR	SPA2	SPA1	LIM1	Q_{ppa}
By the classifiers	SAR2	222	0	0	0	5	98%
	LISR	59	149	0	0	21	65%
	SPA2	0	0	382	162	8	69%
	SPA1	0	0	0	28	0	100%
	LIM1	0	0	3	33	468	93%
	Q_{sens}	79%	100%	99%	13%	93%	

It should be mentioned that validation points do not concern any water. ICU classifies these pixels in the most approximate class (SPA1, pink), but quality of these areas are not relevant (at the south and the right border). SPA2 (red) is a pure class of Spartima maritima and SPA1 contains Spartima in a non-dominant way. The only point where ICU disagrees with the expertise is the small red spot at the right lower corner (instead of pink, see Fig 6). The very narrow resemblance of the spectra of these two classes explains that. Classification is quite globally correct (Kappa-index 0.81 - average accuracy 81.1%, see [16] in section 5.1). Fig 7 shows the map of overlaps, designed to test overlapping rules. The whiter a pixel, the more rules are activated. If no rule can be used, the pixel is shown in red. Even if the map of overlaps is automatically made, it shows pixel classes and the degree of mixing. The image below demonstrates that no knowledge about water evolved.

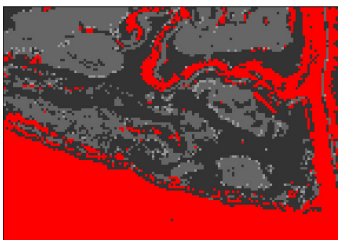

Fig 7: Map of overlaps (ICU classification)

4.3 XCS classification

The following section presents the results obtained by XCS.

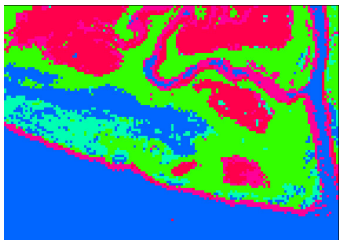

Fig 8 : Classified image

Table 3: Confusion matrix

		Classification by the expert					
		SAR2	LISR	SPA2	SPA1	LIM1	Q_{ppa}
By the classifiers	SAR2	266	12	0	0	3	95%
	LISR	11	133	0	0	2	91%
	SPA2	0	0	353	83	1	81%
	SPA1	0	0	27	106	3	78%
	LIM1	4	4	5	34	493	91%
	Q_{sens}	95%	89%	92%	48%	98%	

The classification with XCS shows quite good results. The same remark should be made about the color of what one believes being a water class, which actually is not relevant. ICU and XCS find the same classes for the same points, including the border of the salt marsh, which contain a lot of mixels. XCS has spent many classifiers to separate SPA1 and SPA2 correctly. Nearly one classifier was used to describe each pixel. Nevertheless, a better global accuracy was obtained (Kappa-index 0.88 - average accuracy 87.7%).

4.4 Conclusion

The case studies have demonstrated the high capacity of the two evolution-based classifier systems to interpret and classify heterogeneous and complex images (e.g. high number of mixels and noisy data). Such hyperspectral images provide in general a computational complexity of $O(n^3)$, which is quite heavy for a deterministic algorithm. It must be noted that the quality of learning is highly related to the quality of the classified image used for rule discovery. The discovered classification rules are accurate enough in terms of representation for ICU and of generalization for XCS to be mutually exclusive and maximally specific. The learning time for the sets of points was relatively short (for a Pentium 2GHz, it lasts 5 minutes for ICU, 15 minutes for XCS). Classified images by the discovered rules have shown that the evolution-based classifier is able to faithfully reproduce the human expertise.

5 Comparison between ICU and XCS

The two algorithms, ICU and XCS are different in many respects. XCS is an online LCS whereas ICU is a bit closer to a genetic algorithm. Consequently, the evolutionary paradigm used, the parameters, and the computation time are not the same. Moreover, the concepts of *pool* are different and serve different purposes. In XCS, the learning takes place in one big pool. In our experiments, classifiers predicting different classes are built together. On the contrary, each class (pool) is kept separated in ICU. The idea is to disallow takeovers of strong rules describing an *easy* class possibly deleting weak rules describing a more complex class. Unfortunately in this setting, the classifiers cannot use previously learned knowledge to improve the current learning process and ICU does not learn a complementary mapping of the environment, as XCS does. However, also XCS may be modularized learning classes separately.

In order to test these questions concerning evolution quality, learning time, and parameter dependency, ICU and XCS were tested in a *validation platform* - a program used to run many tests in a batch way – providing various statistics and curves discussed below.

5.1 Confusion matrices CM and DCM

The goal of this section is to identify basic quality measures that will be used to compare the obtained results with the two systems. Let M_n be a confusion matrix as defined in Table 1.

$$M_n = \begin{bmatrix} m_{1,1} \cdots m_{1,n} \\ \vdots \quad \vdots \\ m_{n,1} \cdots m_{n,n} \end{bmatrix} \quad [11]$$

where $m_{i,j}$ denotes the number of samples classified by the classifier in the class *i*, whereas these samples are for the expert in the class *j*.

The following equations present two common quality measures, global accuracy G_{acc} and the Cohen's Kappa index *K* [Coh60, Lan77].

$$G_{acc} = \frac{\sum_{i=1}^{n} m_{i,j}}{\sum\sum m_{i,j}} \quad [12]$$

$$\delta = \sum m_{i,i} \quad [13]$$

$$\eta = \sum\sum m_{i,j} \quad [14]$$

$$\lambda = \frac{\sum_{c=1}^{n} \left[\sum m_{i,c} \sum m_{c,j} \right]}{\eta} \quad [15]$$

$$K = \frac{\eta\delta - \lambda}{\eta^2 - \lambda} \quad [16]$$

Generally classification problems are a one-to-one classification type: one pixel should be classified by one rule. But the modern conception of ICU or XCS allows solving one-to-*n* classification problems: one pixel can be classified in several classes thanks to the activation of several classifiers. This possibility can bring much of assistance in remote sensing classification problems, for at least two reasons: the presence of *mixels* in the raw data (pixels of several cover types due to the low resolution of sensor detectors) and the fact that human experts produce mixed expertise by labeling the same pixel with different classes, labeling each class by different proportions. Even if ICU and XCS were designed to solve one-to-*n* problems, sections 3.2.4 and 3.3.3 presents different methods to select one rule by a determinist mean if the action set has several of them. The fact, using these methods for the classification of a test set allows filling a confusion matrix (*CM*). But the results obtained from the pool can directly be exploited to take into consideration all expert

information, or to simply investigate the contribution of the selection methods in terms of classification accuracy. Direct-confusion matrices (*DCM*) were designed in this spirit. Whereas the sum of each column of a confusion matrix must reach 100%, different behaviors can occur for the DCM. Note that DCM constitutes a test more difficult to pass for the two methods. All the quality measures (G_{acc}, K) can be applied on confusion matrices or on direct-confusion matrices.

5.2 Quality of the evolution

XCS comes with a lot of parameters and values acquired during the learning. Fig 9 presents the percentage of correct rules obtained at a given generation. A generation is a step of the genetic algorithm. Each step represents a *trial* that is a problem presented to the algorithm. These steps are alternatively an exploration or exploitation. In an exploration step, action could be chosen more randomly and a discovery component [Wilson, 1995] is run in which crossover or mutation can occur. The number of generations (G_{XCS}) is an important parameter. If it is too low, the algorithm does not have the time to learn and the quality of the rules is too bad. If it is too high, learning time is wasted and, despite the continuous generalization in XCS [Butz, 2004], overspecializations may occur. The percentage of correct rules grows quite fast but never reaches a value larger than 0.98: new rules are added in the population. The graph shows a constant and controlled renewal of the population.

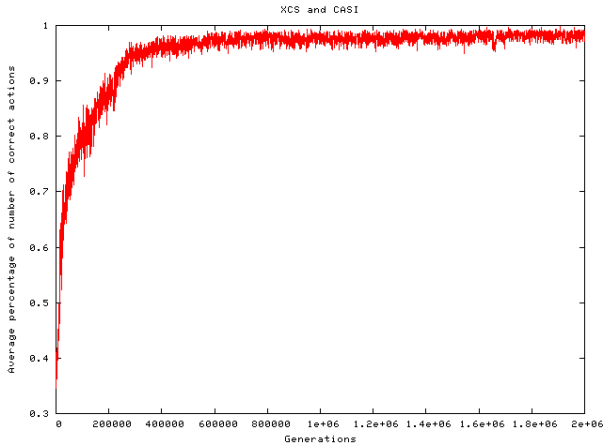

Fig 9 : Percentage of correct classifier depending on the number of generations

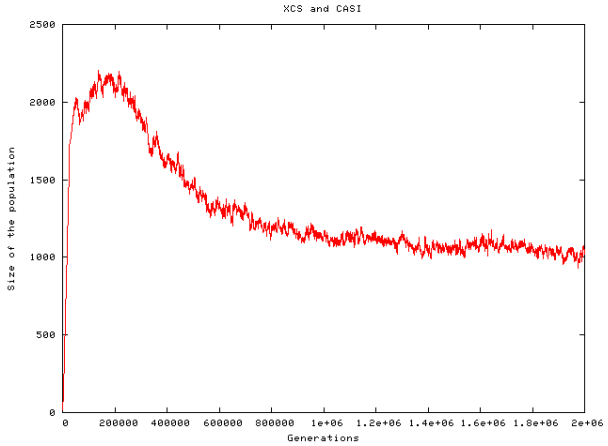

Fig 10 : Size of the population depending on the number of generations. Each single classifier is counted once, some either its numerosity [Wilson, 1995]

In XCS a macro-classifier component [Wilson, 1995] is implemented, allowing a new classifier to be added in the population only if it does not already exist (otherwise, a numerosity parameter in the existing classifier is incremented). Fig 10 presents the size of the population (i.e. the number of unique classifier) depending on the number of generations. A lot of specific rules were generated in the beginning of the learning, each one coding for one or two pixels, later, more general rules are discovered deleting the specific ones effectively

reducing the size of the pool. Depending on the validation method (simple test or k-fold cross-validation), between 500 and 750 points were used in the training set. With this point of view, 1000 rules are rather excessive to encode the knowledge. In fact, a lot of rules are similar and could have been eliminated by the macro-classifier component. Additionally, due to the online learning nature of XCS, the distributed representation using 1000 rules may have advantages in flexibility and adaptation if concept classes drift.

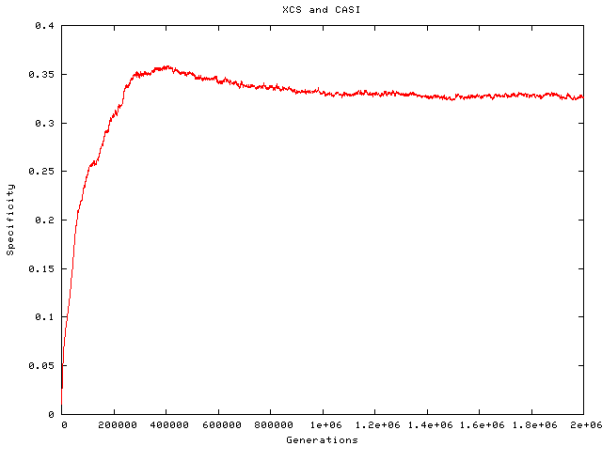

Fig 11 : Specificity of the pool depending on the number of generations

In support with Fig 10, Fig 11 shows the specificity of the rules. The more specific a rule is, the less there will be *jokers* in this rule and the less applicable it will be for new pixels. In the case of hyperspectral data, specificity is measured with respect to the maximal range for the values of the reflectance of the pixels. First of all, the algorithm has no knowledge injecting specific rules for the classification problem. The specificity increases. Overspecializations occur. XCS detects and deletes some of them by its indirect continuous generalization pressure.. This behavior was expected and was observed in all our experiments. In fact, the graph adds some information for choosing a correct number of generations, compared to Fig 9 (400 000 according to Fig 9, but one million by looking Fig 11).

5.3 Learning Time

The learning time is an important issue. It is mainly dependent on two parameters. The points in Fig 12 present the time (in ms) of an XCS-learning (axis Z) according to the size of the population for all the classes (axis X) and the number of generations (axis Y). Points are in red or in green depending on the method used for the deterministic selection of a rule in the action set (see section 3.3.3). The maximum that was observed is 8 millions of milliseconds (2 hours). The size of the set of classifiers and the number of generations can be selected and a compromise between quality and time can be found. However, small population sizes result in poor solution quality.

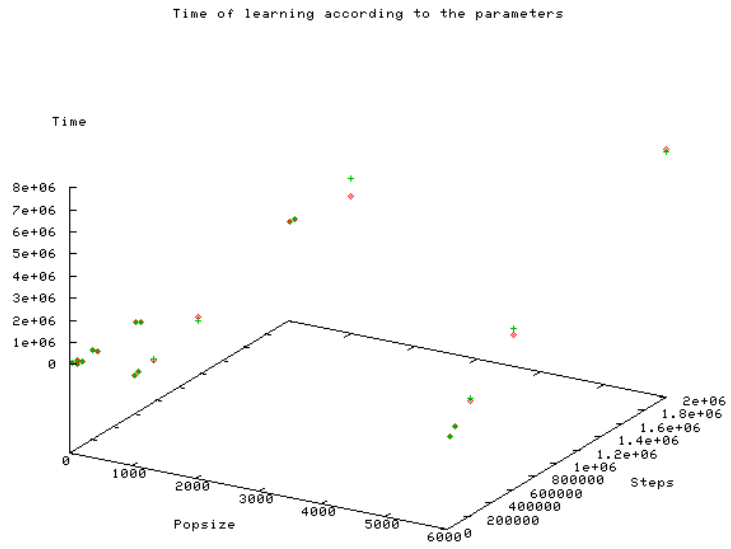

Fig 12 : XCS learning time in ms, depending on the size of the pool (popsize) and the number of generations (steps)

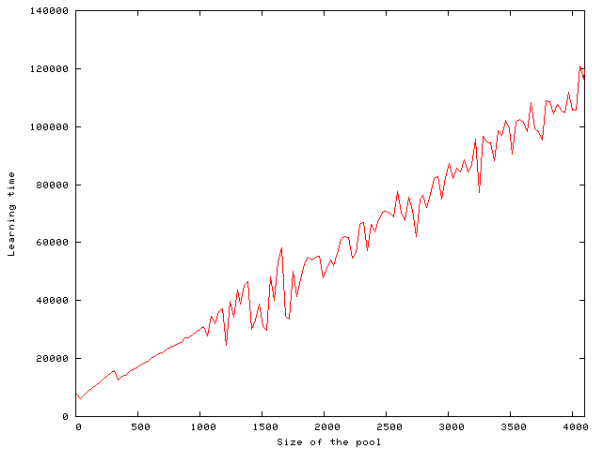

Fig 13 : A projection of the previous graph, the XCS learning time in ms depending on the size of the pool (G_{XCS} = 200000)

5.4 Parameters tuning

It is important that the parameters of genetic evolutionary systems are well-tuned. Fig 13 and Fig 14 was used to set up the size of the pool considering global accuracy at the end of a learning run and the learning time. Size of the pool is a parameter introduced in XCS before the learning to fix the maximum size of the population. A *surround* of the specificity-bump observed in Fig 11 must be accomplished. Between 2000 and 3000 rules are necessary to obtain a good pool (accuracy approximates 0.9), without having to wait a long time.

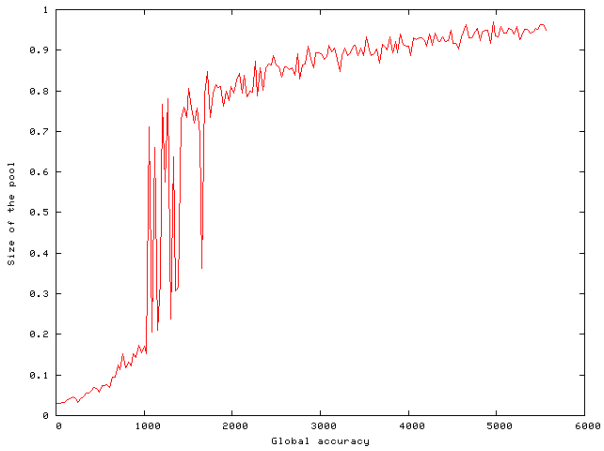

Fig 14 : Final global accuracy obtained with XCS after learning corresponding to given sizes of the pool

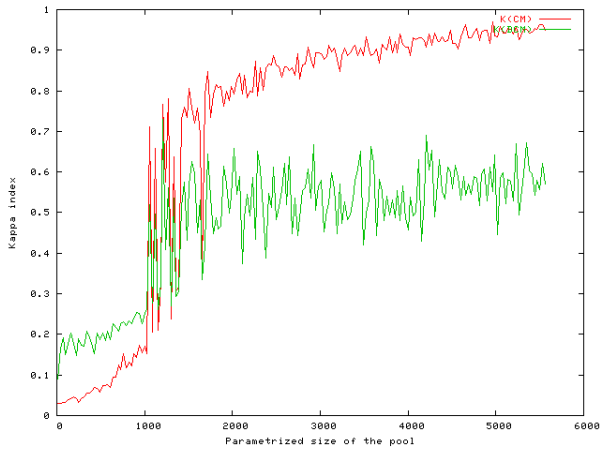

Fig 15 : Kappa index for the confusion matrices (in red) and Kappa index for the direct confusion matrices, obtained with the given sizes of the pool

Accuracy measurement is a rustic average of the diagonal of the confusion matrix obtained on the training set corresponding to each experiment. A more interesting measure is the Kappa index, applied on the confusion matrices (CM) or the direct confusion matrices (DCM) (see section 6.1). Below 1500 rules, the pool is quite unstable and has not enough specific classifiers to allow construction of more generic ones. This can be explained by the large range of values covered by the data. The index never exceeds 0.6 for the DCM, but reaches 1 by looking at the CM. The difference lies in the algorithm selecting the final rule when more than one rule for the same class are kept in the action set. Fig 15 proofs the quality of this algorithm.

Fig 16 and Fig 17 show a comparison between ICU and XCS regarding two essential and common parameters. ICU is presented in blue, XCS in red. The graph shows the quality of the methods, depending on the number of generations and the size of the set of classifiers. Because of the difference between the two methods, the set of parameters S_{ICU} and S_{XCS} was not the same, but the cover went from the small ones to large values from the point of view of the corresponding method. The number of generations has no important effect on the final accuracy. It only guarantees that the pool will not be too specific at the end. The size of the pool is a relevant parameter only for XCS. Below a given threshold-size, ICU is better, above XCS is better.
The compromise between the time of computation and the global accuracy can be found by selecting the method.

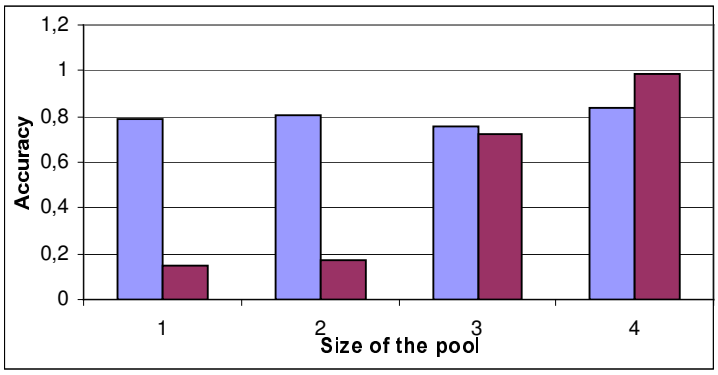

Fig 16 : Final accuracy depending on fixed sizes of the pool. S_{ICU} = 100, 200, 300, 500. S_{XCS} = 20, 100, 1000, 6000

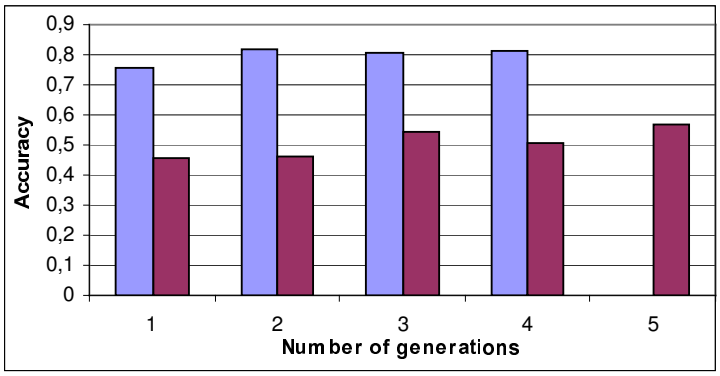

Fig 17 : Final accuracy depending on fixed sizes of the number of generations. G_{ICU} = 100, 200, 300, 500. G_{XCS} = 20000, 60000, 200000, 600000, 2000000

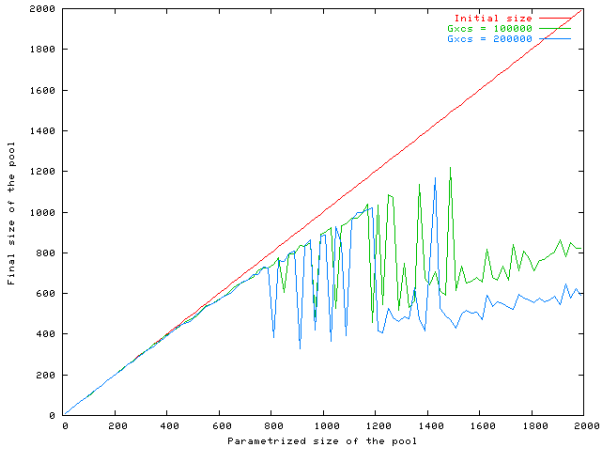

Fig 18 : Sizes of the pool at the end of the learning, C_v = 1

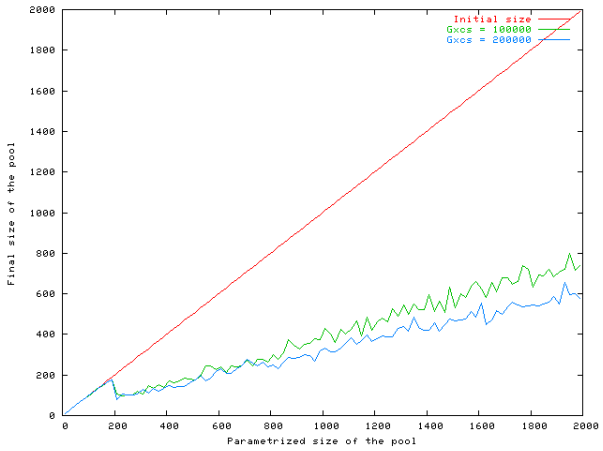

Fig 19 : Sizes of the pool at the end of the learning, $C_v = 10^6$

These two last graphs (Fig 18 and Fig 19) have shown the factor reduction between the size of the population fixed at the beginning of each experiment and the final size after the reduction obtained from the macro-classifier and the generalization component. The fixed initial size counting the numerosity of each classifiers is printed in red, in blue the final size obtained with a 200000 generations learning and in green the final size obtained with a 100000 generations learning. The only difference between Fig 18 and Fig 19 resides in the cover-rand parameter (see section 3.3). In Fig 18, a low cover-rand ($C_v = 1$) implies a high specificity and thus diversity in the initial population. On the other hand, In Fig 19, a high cover-rand ($C_v = 10^6$) operator implies high generality and thus lower diversity in the initial population. Interestingly, the final pool size differs dependent on the initial pool size. This behavior can be explained by XCS's learning nature: the larger the maximum population size N, the more diversity can be supported in the population. Thus, a very low N prevents diversity and possibly disrupts the accurate learning of all classes. On the other hand, a large N allows high diversity and thus generates many new and/or irrelevant rules.

5.5 ROC curves

ROC curves are a well-known statistical measure of the classification quality based on the concepts of specificity and sensitivity. Specificity is the true negative rate of classification and sensitivity is the true positive rate of a classification. Each curve is usually associated with a parameter *p*. A new pool of rules $P_p = \Phi(P_0, p)$ was generated from an initial pool P_0 (corresponding to a reference pool trained with the training set) and was applied on the testing set. In our case, *p* corresponds to a variation coefficient for the definition range of the rule ($p \in [0;10]$). After the application of this coefficient, the condition part $[m_i^R; M_i^R]$ of the rule *R* for the attribute *i* is replaced by $[\mu - \varepsilon; \mu + \varepsilon]$ with $\mu = \frac{m_i^R + M_i^R}{2}$ and $\varepsilon = \mu.p$. The hypothesis behind this study is that the spectra of each sample in the testing set are uniformly distributed between the borders of the majority of the condition parts. This hypothesis has been verified on P_0 for the majority of points of the training and testing set.

Fig 20 and Fig 21 show the ROC graphs for XCS and for ICU. One can see in Fig 20 that the class C_5 (in green) gives many false alarms. With ICU, all the classes are globally correct, except the class C_4 (in purple). Clearly, an interaction between learning method and data type is observed: if one looks at C_4, for low values of *p*, ICU gives better sensitivity and specificity (less False Negatives or False Positives) than XCS and could be useful for diagnoses (better results if the test is targeted for a given class). Conversely, XCS gives better sensitivity and specificity for high values of *p* and could be useful for tracking down, because the accuracy is better even if the sensitivity of the test increases. The ROC curves are a useful tool to choose the rule discovery method according to the class that we want to classify.

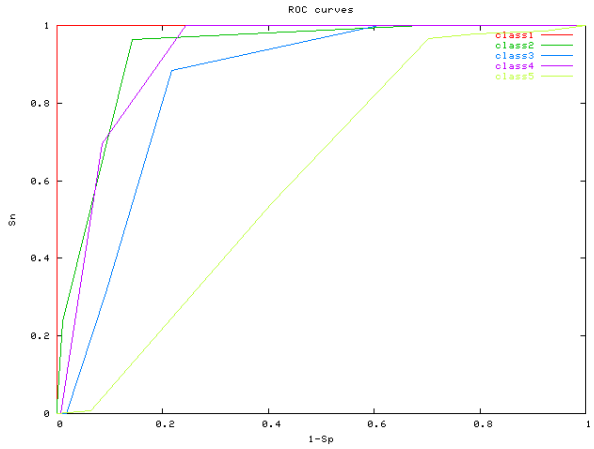

Fig 20 : XCS ROC curves for each class, obtained on CASI data

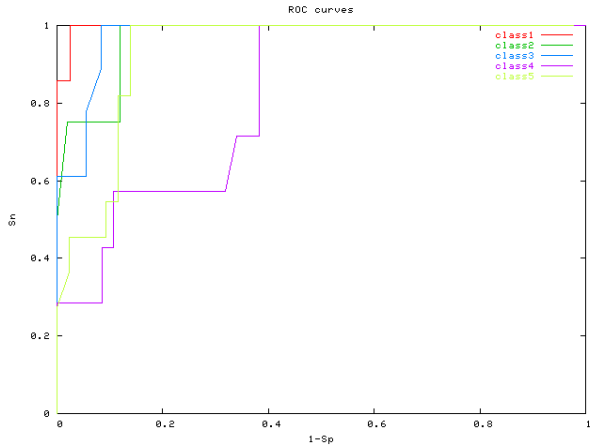

Fig 21 : ICU ROC curves for each class, obtained on CASI data

5.6 Validation strategies

Many validation strategies were used to give an accurate comparison between the results of ICU and XCS, in terms of obtained confusion matrices. Simple hold-one-out strategy, leave-1-out k-fold cross-validation, jack-knifing and bootstrapping strategies [Hjorth, 1994, Plutowski, 1994, Shao, 1995] were used. In this paper, only cross-validation statistics have been examined. Cross-validation consists to cut the set of points in k parts randomly. Then the testing is applied on each part and the learning on the remainder.

Table 4(a) shows the results for ICU. The column *Mean* is the average of the results for the k parts and the standard deviation (*E-Type*) indicates the variation in the quality of each part. Correct accuracy and Kappa index were obtained for the confusion matrices. For different values of k, same results were obtained. In average for ICU, we obtained an accuracy of 0.81, and this value is quite representative of its quality for these sorts of tests. Table 4(b) shows the results for XCS. The results are better than ICU. However, the variations of quality between experiments are larger due to the high level of stochastic behavior in the XCS method.

Sometimes the rules overlap themselves because the same pixel can activate several rules. Kappa-DCM was designed to test this overlap. In fact, bad results might be produced but the algorithm that finally chooses the appropriate rule improves the results.

Table 4 : k-fold leave-1-out cross-validation results

		ICU (a)		XCS (b)	
Test	k	Mean	E-Type	Mean	E-Type
Accuracy	3	0.821	0.007	0.852	0.037
	5	0.794	0.026	0.903	0.026
Kappa CM	3	0.821	0.007	0.852	0.037
	5	0.794	0.026	0.903	0.026
Kappa DCM	3	0.777	0.023	0.417	0.007
	5	0.648	0.017	0.536	0.015

6 Conclusions and Perspectives

Until now, the applications of the evolutionary methods in the domain of remote sensing image analysis are still very rare. A few recent reports have shown interesting results using genetic programming approaches [Ross, 2002; Brumby 2002]. One of the main objectives of this study was to introduce and provide a better understanding of learning classifier systems applied to remote sensing data.

This report has drawn a comparison between two classifier systems, ICU and XCS, applied to voluminous remote sensing data. ICU discovers an *if … then* classification rule for each class using a fitness function based on image classification quality. In general, these rules are robust and can manage several disjunctions of constraints. They are able to describe the complexity of the analyzed spectra. XCS learns a pool of classification rules for all the classes. These rules were proven to improve the accuracy of the expert and are sufficiently generic for applying them on other portions of satellite images.

Compared to XCS, the knowledge learned by ICU is not distributed in a pool, but the representation of the rules is more complex and the evaluation of new data is faster. Conversely, XCS can learn a complete mapping of the environment: the direct relation between a pixel and its class is learned in the same time as the opposite relation (a rule saying that this pixel is *not* in the class). Sometimes the learning of the opposite relation is more relevant and compact than for the relation itself. This approach appears very useful to depict non-compact classes, very distributed in the search space, as produced sometimes by unsupervised learning. This difference in design has affected the results of tests according to a given criterion obtained in the section 6. The global accuracy of XCS was better than ICU (in the order of five to seven percents), but ICU runs faster (during the training and the exploitation of the pool) and learns a smaller set of rules: only one rule is needed for each class while XCS needs 1000 rules to learn 250 to 500 points. In terms of knowledge representation, rules in XCS consist of a conjunction of intervals whereas a rule in ICU can be more complex consisting of a disjunction of conjunctions of intervals. The selection methods described in the paper were experimented on the CASI set, showing slightly better accuracies obtained with the *CloseCenter* method for ICU and the *ScoredConfident* method for XCS.

Taking into consideration image complexity and noisy data, the results of evolutionary methods in our experiments are very encouraging. Case studies have demonstrated that the obtained rules are able to reproduce faithfully the terrain reality. The redundant information or the noisy bands have been successfully identified by each pool representation. The representation of the rules has allowed for the modeling of constraints adapted to the granularity of spectral reflectance. Hence, these rules-based approaches are of great interest when compared to traditional methods of classification. But the potential of evolution-based algorithms in remote sensing image classification is just beginning to be explored. Further investigation of ICU and XCS learning efficiency are necessary. For instance, research about post-processing the rule base represents a great challenge for XCS and more powerful representation of the rules in ICU including spatial knowledge and constraints adapted to temporal series could give better results and yield more relevant rules.

References

[Bernadò, 2002] Bernadò E., Llorà X., Garrel J. M., XCS and GALE: A comparative study of two learning classifier systems and sic other learning algorithms on classification tasks. [in] Lanzi, P. L., Stolzmann, W., Wilson, S. W. (Eds.), Advances in Learning Classifier Systems (LNAI 2321), 115-132, 2002.

[Bernadò, 2003] Bernadò E., Garrel J. M., Accuracy-based learning classifier systems: Models, analysis, and applications to classification tasks. *Evolutionary Computation*, 11, 209-238, 2003.

[Bock, 1999] Bock H. H., Diday E. (eds.), Analysis of Symbolic Data. Exploratory Methods for Extracting Statistical Information from Complex Data, [in] Studies in Classification, Data Analysis and Knowledge Organization, vol. 15, Springer-Verlag, Heidelberg, 1999.

[Bonarini, 2000] Bonarini A., An Introduction to Learning Fuzzy Classifier Systems, [in] P-L. Lanzi, W. Stolzmann & S.W. Wilson (eds) Learning Classifier Systems: From Foundations to Applications. Springer, pp 83-106., 2000.

[Brumby, 2002] Brumby S. P., Pope P. A., Galbraith A. E., Szymanski J. J., Evolving Feature Extraction Algorithms for Spatio-Spectral Remote Sensing of Vegetation. *Proceedings of Spectral Remote Sensing of Vegetation Conference*, 2002.

[Butz, 2002] Butz M. V., Wilson S. W., An algorithmic description of XCS. *Soft Computing*, 6, 144-153, 2002.

[Butz, 2003] Butz M. V., Sastry K., Goldberg D. E., Tournament selection in XCS. *Proceedings of the Fifth Genetic and Evolutionary Computation Conference (GECCO-2003)*, 1857-1869, 2003.

[Butz, 2004] Butz M. V., Kovacs T., Lanzi P. L., Wilson S. W., Toward a theory of generalization and learning in XCS, *IEEE Transactions on Evolutionary Computation*, 8: 28-46, 2004.

[Cohen, 1960] Cohen J., A coefficient of agreement for nominal scales. Educ Psychol Meas, 20:37-46, 1960.

[DAIS, 2001] Proceedings of the Final Results Workshop on DAISEX (Digital AIrborne Spectrometer EXperiment), ESTEC, Noordwijk, 2001.

[Dave, 2002] Dave V., Tim H., How To Lie With An Error Matrix, Remote Sensing & Image analysis (online paper), 2002.

[Dixon, 2002] Dixon P. W., Corne D. W., Oates M. J., A preliminary investigation of modified XCS as a generic data mining tool [in] Lanzi, P.L., Stolzmann, W., Wilson, S.W. (Eds.), Advances in Learning Classifier Systems (LNAI 2321), 133-150, 2002.

[Hjorth, 1994] Hjorth J. S. U., Computer Intensive Statistical Methods Validation, Model Selection, and Bootstrap, London, Chapman & Hall, 1994.

[Horn, 1994] Horn J., Goldberg D. E., Deb K., Implicit niching in a learning classifier system: nature's way, [in] Evolutionary Computation, 1994.

[Kallel, 1997] Kallel L., Schoenauer M., Alternative random initialization in genetic algorithms, [in] Proc. of ICGA'97, Morgan Kaufmann, San Fransisco, pp. 268-275, 1997.

[Korczak, 2003a] Korczak J., Quirin A., Evolutionary Mining for Image Classification Rules, 6th International Conference on Artificial Evolution (EA2003), Marseille, 2003.

[Korczak, 2003b] Korczak J., Quirin A., Evolutionary Approach to Discovery of Classification Rules from Remote Sensing Images, 5th European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP2003), Essex, 2003.

[Landis, 1977] Landis J. R., Koch G. G., The measurement of observer agreement for categorical data. Biometrics 1977 Mar, 33(1):159-174, 1977.

[Lanzi, 1997] Lanzi P. L., A study of the generalization capabilities of XCS [in] T. Baeck, ed., Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97), Morgan Kaufmann, 1997.

[Lanzi, 1999] Lanzi P. L., Extending the Representation of Classifier Conditions., Part II: From Messy Coding to S-Expressions [in] Banzhaf et al., pp 345-352, 1999.

[Plutowski, 1994] Plutowski M., Sakata S., White H., Cross-validation estimates IMSE [in] Advances in Neural Information Processing Systems 6, pp. 391-398, 1994.

[Quirin, 2002] Quirin A., Découverte de règles de classification : classifieurs évolutifs, Mémoire DEA d'Informatique, Université Louis Pasteur, LSIIT UMR-7005 CNRS, Strasbourg, 2002.

[Ross, 2002] Ross B. J., Gualtieri A. G., Fueten F., Budkewitsch, P., Hyperspectral Image Analysis Using Genetic Programming. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), pp. 1196-1203, 2002.

[Shao, 1995] Shao J., Tu D., The Jackknife and Bootstrap, New York, Springer-Verlag, 1995.

[Stow, 2004] Stow et al., Remote sensing of vegetation and land-cover change in Arctic Tundra Ecosystems, Remote Sensing Environment, 89: 281–308, 2004.

[TIDE, 2001] Tidal Inlets Dynamics and Environment, research project supported by the European Commission under the Fifth Framework Programme, contract n° EVK3-CT-2001-00064, http://www.istitutoveneto.it/tide, 2001-2005.

[Toutin, 2002] Toutin T., Cheng P., Quickbird – a milestone for highresolution Mapping. Earth Observation Magazine, 11(4):14-18, 2002.

[Webec, 1995] Weber C., Images satellitaires et milieu urbain, Hermès, Paris, 1995.

[Wilson, 1994] Wilson S., ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2(1), 1-18, 1994.

[Wilson, 1995] Wilson S., Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2), 149—175, 1995.

[Wilson, 1998] Wilson S., Generalization in the XCS classifier system. *Proceedings of the Third Annual Genetic Programming Conference*, 665-674, 1998.

[Wilson, 2000] Wilson S., Get Real! XCS with Continuous-Valued Inputs [in] P. L. Lanzi, W. Stolzmann, & S.W. Wilson (Eds.), *Learning Classifier Systems (LNAI 1813)*, 209-219, 2000.