3 Discovering of Classification Rules from Hyperspectral Images

Arnaud Quirin and Jerzy Korczak

3.1. 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 data analysis. At the same time, a lot of useless information can hide valuable information. These observations force classification systems 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 remote sensing images using traditional and statistical techniques such as Support Vector Machines [7] or neural networks [16, 21, 22]. But, to our knowledge, relatively few researchers in the evolutionary community have considered how classification rules might be discovered from raw and expertly classified images. Only some works have been done using genetic programming approach [15, 28], but no papers have been published about the effectiveness of learning classifier systems in this field[1]. To discover classification rules the unique source of information is a remote sensing image and its corresponding identification furnished by an expert. Generally the images, registered by various satellites (e.g. SPOT, CASI, Quick Bird), 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 [18, 24, 32]. This chapter presents the potential contribution of evolutionary-based techniques to discover the rules. Learning classifier

[1]This review concerns the main conference on learning classifier systems, *IWLCS* (International Workshop on Learning Classifier Systems) from 1992 to 2005

systems can produce accurate, robust and maximally specific classification rules able to deal with the noisy artifacts contained in remote sensing images.

The aim of this chapter is to describe a process of design and validation of evolutionary classifiers applied to remote sensing images. This system is data-driven because it generates classification rules able to adapt themselves according to the available data and expertises, and it distributes the quantity of rules in an optimal way to describe each class according to the complexity of the data. In general, classification rules are discovered from the established classifier system [12, 35]. In remote sensing, the initial population of classifiers is randomly created from images and given classes, and then evolved by a genetic algorithm until the acceptable solution is found.

In remote sensing literature, several classification approaches are presented, namely:

- pixel-by-pixel: each image pixel is analyzed independently of the others according to its spectral characteristic [14],
- zone-by-zone: before classification, the pixels are aggregated into zones, the algorithms detect the borders of the zones, delimit them by their texture, their repetitive motives [19],
- by object: this is the highest level of recognition, the algorithms classify semantic objects, the algorithms detect their forms, geometrical properties, spatio-temporal relations using the background domain knowledge [17].

Our approach is based on spectral data of pixels; therefore, discovered classification rules are only able to find spectral classes rather than semantic ones. This spectral component of class description is essential to well-recognized thematic classes. It should be noted that the proposed classifier system may be easily adapted to more sophisticated object representations.

To validate our approach a system called *I See You* (ICU) has been used. In the ICU, we have adapted and extended ideas developed in the well-known classifier systems such as XCS [35], the S-classifiers, and "Fuzzy To Classify System" [25]. We have also been inspired by the works of Riolo [27] on gratification and penalization, and of Richards [26] on the exploration of the space of classifiers. To demonstrate the performance of our classifier system, the ICU has been compared with XCS-R and two popular methods, one based on neural networks and the other based on Support Vector Machines [7, 16]. XCS-R is a system based on XCS, integrating the concept of continuous values [36]. XCS is a learning classifier system, developed by [33, 34], that evolves a rule set online based on prediction accuracy and a niched genetic reproduction [20]. The classification systems have been tested in the framework of the European TIDE project [3] on hyperspectral remote sensing images covering the region on Venice.

The chapter is structured as follows. The basic terms and properties of hyperspectral images are introduced in the section 2. The section 3 gives general ideas about what is a « good » classifier system in remote sensing. The section 4 describes our algorithm ICU. In this section, the main components and the quality

measures of the method are explained. The section 5 draws the main principles of XCS-R and the adjustments which were made to make it able to deal with these data. The algorithms are evaluated on remote sensing images covering the region on Venice in the framework of the European TIDE project [3]. The results of the comparison between ICU, XCS-R and two statistical methods (SVM and NN), are presented in section 6. And finally, section 7 concludes the experimentations and indicates the perspectives of the future research.

3.2. Hyperspectral remote sensing images

A 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 [2]). Conversely, the term *multispectral* is used in the case of a low number of spectral bands, as CASI and Quick Bird remote sensors [30]. 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. Figure 3.1 shows the spectrum of reflectance of a pixel from a hyperspectral sensor (type MIVIS). Each spectral channel has roughly 10 nanometers in width.

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 figure 3.2).

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

CASI data: Generated by the airborne spectrometer CASI (Compact Airborne Spectrographic Imager), with a size of 1175x673 pixels, 288 spectral channels (412 - 957 nm), and a high resolution (1.3m). This image has been pre-processed by geometric correction and warping (specifically, first order polynomial warping and nearest neighbor re-sampling).

MIVIS data: Generated by the MIVIS sensor (Multispectral Infrared and Visible Imaging Spectrometer) embarked on a satellite, with a size of 397x171 pixels, 92 to 102 spectral channels (433 - 2478 nm), and a high resolution (2.6m). Same pre-processing has been applied as before.

Learning and testing were applied on subsets of these images. Subsets contained 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. Figure 3.3 shows Quick Bird data for the Lagoon of Venice and the corresponding set of validated points. The rectangle is the area in

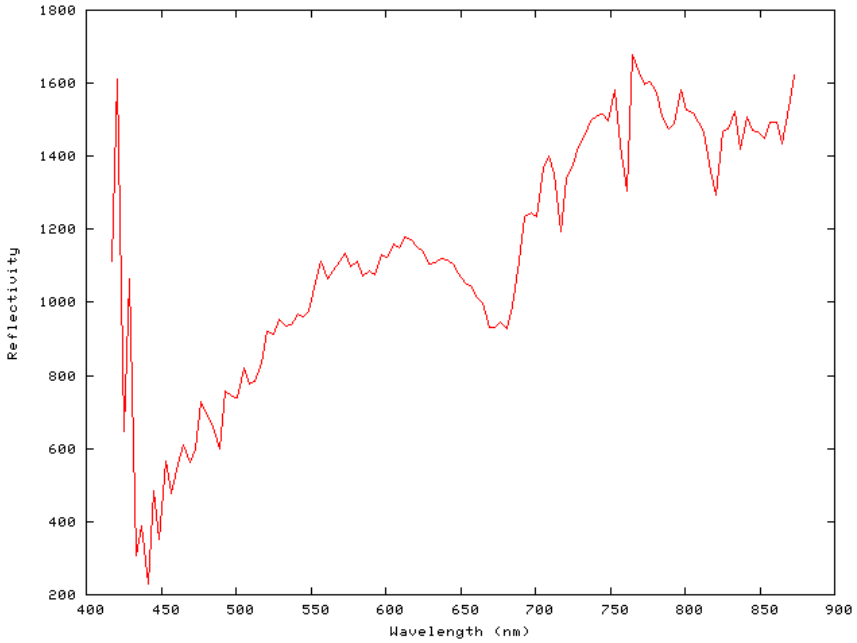


Figure 3.1. Spectrum of reflectance observed for a pixel from the hyperspectral sensor MIVIS

Figure 3.2. Typical noisy channels : respectively Strasbourg (band number 42 and number 59 from DAIS) and Lagoon of Venice (band 45 from DAIS)

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%.

3.3. Definition of a system of classifiers in remote sensing

Generally speaking, a system of classifiers integrates symbolic learning and evolution based computing. Classification rules are symbolic expressions and describe

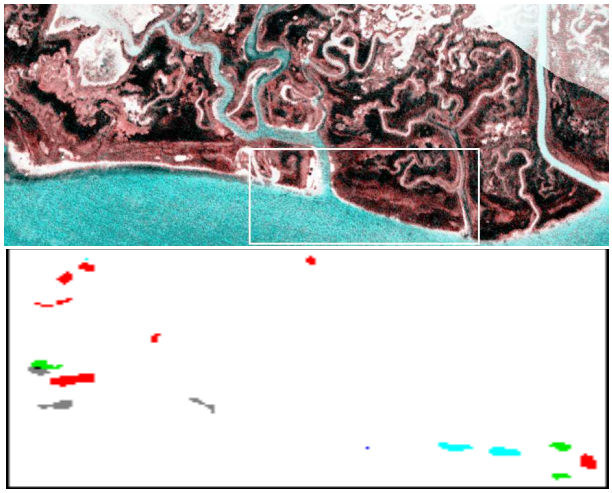


Figure 3.3. Quick Bird data of the Lagoon of Venice and the corresponding set of validated points (ground truthing)

conditions to be held and actions to be taken if the conditions are satisfied. Quality of the rules is evaluated according to their classification performance. Here, we must underscore the fact that the rules are not introduced by a programmer or by an expert.

A system of classifiers is called evolutionary if it is able to adapt itself to the environment. This means that it can modify its knowledge and its behaviour according to the situation. For example, in remote sensing, the size of the classes may evolve in one of two ways: (1) if, after an initial classification, there remain non-classified pixels, or (2) if there are pixels belonging to several classes (mixed pixels). When a classifier integrates one of these pixels to one or another class, it is necessary to dynamically adapt classification rules. In this way, certain rules that treat only the simple cases (not mixed pixels) will become useless, and new rules are necessarily created for other cases.

From a functional point of view, a classifier can be defined as a rule representing a piece of knowledge about a class, and may be a conditional expression, such as *if* <*conditions*> *then* <*action*>. In the early classifier systems [35], each part of a rule was a binary message, encoding elementary information such as a value, colour, form, shape, etc. The "*conditions*" part described an entry message in the system, corresponding to conditions that must be fulfilled in order to activate this rule. The "*action*" part defined the action to be carried out when the appropriate conditions were satisfied. This binary encoding scheme is not well adapted to image classification rules.

One of the reasons for this is based on the domain of spectral values that may be assigned to a pixel (from 0 to 255 for 8-bit pixels, or from 0 to 65000 for 16-bit pixels). Of course, binary encoding of rule conditions is possible but the rules would be difficult to understand. Instead, we assert that the evolved rules

must 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 maximally specific generalizations, meaning that they have to cover the maximum pixels belonging to a given class and the minimum pixels belonging to another classes.

Before rule specification is explained, recall that a pixel is encoded as a spectral vector, defining a value of reflectance for the n bands of the remote sensing image:

$$< \text{pixel} >:= [b_1, b_2, \ldots, b_n] \tag{3.1}$$

In our system, the condition for 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 the pixels defined on 16 bits). This solution allows to partition the space of the spectral values in two ranges: the first containing the pixel values which corresponds 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 $\mathbb{R}^n$ space:

$$< \text{condition} >:= \bigwedge_{i=1}^{n} \bigvee_{j=1}^{k} m_i^j \leq b_i \leq M_i^j \tag{3.2}$$

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 intervals $[m_i^j; M_i^j]$ are not necessarily disjunctive. By experiments, we have found that if we allow the genetic algorithm to create non-disjunctive intervals, instead of merging them, the results of genetic operators are more interesting. We have noticed that merging intervals significantly diminishes the number of intervals, and in the same time reduces the possibilities to create more efficient rules. The example below illustrates a concept of interval merging: $E = [11; 105]$ or $[138; 209]$ or $[93; 208]$ corresponds after merge operation to $E = [11; 209]$.

To satisfy a rule, a pixel has to match at least one spectral interval for each band. Logically speaking, to associate a pixel to a class, its values have to satisfy the conjunction of disjunctions of intervals that define a condition part of the classification rule. Figure 3.4 illustrates an example of matching of two pixels against the spectral class. The left figure shows graphically the spectral intervals of the class defined by a given rule. The next two diagrams show spectral signatures of

two pixels: the first matches the rule, but the second does not. Hence, only the first pixel of this example may be considered to be an instance of the class.

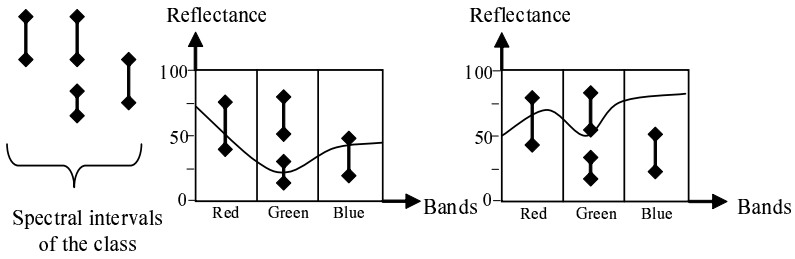


Figure 3.4. Matching spectral bands and spectral signature of pixels. For instance, a real-image sample for the rule could be (see the figure): « *if* $40 < R < 80$ AND $(20 < G < 30$ OR $55 < G < 85)$ AND $25 < B < 50$ *then class X* », where R, G and B are the channels of a three-bands captor.

This representation of the rule has been chosen mainly because of its simplicity, compactness and uniform encoding of spectral constraints. During experimentation, this representation has demonstrated rapid execution of genetic operators and efficient computing. Of course, one may specify more complex structures using spatial properties of the pixel, with respect to the pixel neighbourhood. Also, one may include features resulting from thematic indices or mathematical operators applied to pixel environment. We may also apply a genetic programming to identify new characteristics. These semantically extended formalisms are interesting, however they not only require more sophisticated genetic operators, but also more powerful computers to perform the calculation in an acceptable amount of time.

3.4. From the rule creation to the evolution with ICU

3.4.1. Genetic algorithm

To discover a classification rule, ICU uses a Michigan-like learning classifier system representation, in which each rule is encoded by one individual [24]. 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. Initial classification rules are created based on the extreme maximum and minimum values for defined spectral intervals, taking into account every class. It should be noted that by this initialization, rule searching is considerably reduced, and initial intervals are very close to the final solution. During this process, the initial spectral limits are slightly perturbed by adding a random value to lower and upper spectral limits. Hence, the initial population of classification rules is quite diversified.

This initial pool of classifiers is evolved from a genetic algorithm. Our system searches for a best classifier for each class, independently. A major reason for

choosing this procedure is the efficiency of computations; that is, the process of rule discovery is not perturbed by other rules.

The quality of classification rules is based on a comparison of these results with the image classified by an expert. If pixels covered by the classifier perfectly overlap those indicated by an expert, then the system assigns the highest quality value to the classifier; otherwise, in the case of some mismatching, the quality factor is reduced (between 0 and 1). An associated fitness function will be detailed in the next section. During the evolution process, the rules are selected for the crossover according to the quality for a given class. The process of rule evolution is defined in the algorithm A1 below.

ALGORITHM A1
PROCESS OF RULE DISCOVERY

{R is a classification rule and P_0, P_1 and P_2 populations of classification rules}
R := INITIAL_RULE*(images)* *// Creation of a rule according to spectral extremes*
P_0 := INITIALIZATION*(R)* *// Random perturbation of rules*
EVALUATION*(P_0)* *// Calculation of the fitness function for each rule*
while TERMINATION_CRITERION*(P_0) = false* ***do***
  P_1 : = SELECTION_X*(P_0)* *// Selection for crossover*
  P_1 : = CROSSOVER*(P_1)* $\cup$ COPY*(P_0)*
  P_2 : = SELECTION_MUT*(P_1)* *// Selection for mutation*
  P_2 : = MUTATION*(P_2)* $\cup$ COPY*(P_1)*
  EVALUATION*(P_2)*
  P_0: = REPLACEMENT*(P_0,P_2)* *// New generation of rules*
end while
Result: R, the classification rule for a given class

As mentioned before, this algorithm must be designed to run independently for each class. This allows for obtaining classifiers according to user requirements without the necessity of carrying out computations for all classes with the same level of quality. This also allows for the maintenance of previously generated classifiers, as well as for the introduction of new ones. Further, the user may define a hierarchy of classes and specialize some classifiers while respecting newly created sub-classes with different levels of classification quality.

3.4.2. The evaluation function

The evaluation function serves to differentiate the quality of generated rules and guide genetic evolution. Usually, this function depends strongly on application domain. In our work, evaluation is based on the classification obtained by the

classifier (I_{rule}) and the expertly given classification (I_{expert}). Table 3.1 defines the values necessary to compute the evaluation function.

		Image classified by the classifier R (I_{rule})	
		No. of pixels activating R	... non activating R
Pixel classified by the expert E (I_{expert})	True	P_{exp}^{rul}	$P_{exp}^{\overline{rul}}$
	False	$P_{\overline{exp}}^{rul}$	$P_{\overline{exp}}^{\overline{rul}}$

Table 3.1. Characteristics of the evaluation function

In a given system, the evaluation function is computed as a balanced score between sensitivity and specificity, two popular quality measures in remote sensing:

$$N_{final} = \alpha N_{class} + (1-\alpha) N_{\overline{class}} \tag{3.3}$$

where $N_{class} = \frac{P_{exp}^{rul}}{P_{exp}^{rul} + P_{exp}^{\overline{rul}}}$ is the sensitivity and $N_{\overline{class}} = \frac{P_{\overline{exp}}^{\overline{rul}}}{P_{\overline{exp}}^{\overline{rul}} + P_{\overline{exp}}^{rul}}$ is the specificity. α is called the adjusting coefficient, which is used for certain classes that are under- or over-represented. By default, the value of this coefficient is equal to 0.5.

The proposed function has a number of advantages; it is independent of the pixel processing sequence, invariant of the size of classes, and efficient for class discovery with a highly variable number of pixels.

The evolution process converges according to some statistical criteria indicating if the current classifier is near to a global optimum or if the population of rules will not evolve anymore. The termination criterion of the algorithm leans on the statistics of classifier quality evolution. In our system, we take into consideration not only the evolution of quality of the best and the average classifiers, but also the minimum acceptable quality defined by a user and a maximal number of generations to run. If one of these criteria is satisfied, then the process is stopped.

The most difficult determination to make is whether the quality of a classifier is not continuing to evolve. To detect stabilization of the quality measure, we have based our heuristics on statistics regarding quality evolution of the best classifier. For example, let Q_k be the quality of the best classifier obtained during the last k generations, and Q_o be the quality of the best classifier of the current generation. The algorithm is interrupted if the following equation is satisfied:

$$\left| \frac{\sum_{k=1}^{P} Q_k}{P} - Q_o \right| \leq E \tag{3.4}$$

where P represents the maximum period of quality stabilization, and E is a maximal variation of this stabilization compared with the current quality.

It is important to have an initial population of classifiers within the vicinity of the solution to be found. Two algorithms have been proposed allowing for the generation of a diversified pool of classifiers close to the expert hidden classification rule. The first, called *GenMinMax*, creates maximum intervals covering the expert rule, and the second algorithm, called *GenSpectro*, integrates the spectral distribution density and interval partitioning [24].

3.4.3. Genetic operators

One of the most important tasks while designing a genetic algorithm is to invent operators that will create new potential solutions. All of our operators have been specialized on classifier representation, and they have been validated on remote sensing images. With respect to software engineering, the genetic algorithm has been structured into layers corresponding to genetic operations (e.g. selection, mutation, crossover and replacement). The system is viewed as a collection of layers with data passed from layer to layer. Layer execution follows from one to another, and genetic operations are invoked in the same sequence. This modular approach makes program maintenance and future extensions much easier.

Selection of classifiers. In general, selection is the operation of allocating reproductive opportunities to each classifier. The reproductive force of a classifier is expressed by a fitness function that measures the relative quality of a classifier by comparing it to other classifiers in the population. There are many methods for selecting a classifier [6]. In our system, the selection operator is applied in the following cases:

- choosing the classifier to be reproduced for crossing, or muting;
- repetition of the classifier, depending on whether it completes the genetic pool after having completed the crossover;
- preservation of a classifier from the former genetic pool for the next generation;
- elimination of a classifier in a newly created genetic pool based on an assigned rank.

Selection strategies are well known: the roulette wheel, ranking, elitism, random selection, the so-called tournament, and eugenic selection. Our experiments have shown that roulette wheel selection is most advantageous for the reproductive phase, but the tournament strategy with elitism is best for the generational replacement scheme.

Crossover of classifiers. Crossover requires two classifiers, and cuts their chromosome at some randomly chosen positions to produce two offspring. The two new classifiers inherit some rule conditions from each parent-classifier. A crossover operator is used in order to exploit the qualities of the classifiers.

Each result of the crossover process has to be validated. Validation of the various rule attributes (border limits violation, overpasses, etc) is carried out by a process of interval merging, as shown in figure 3.5.

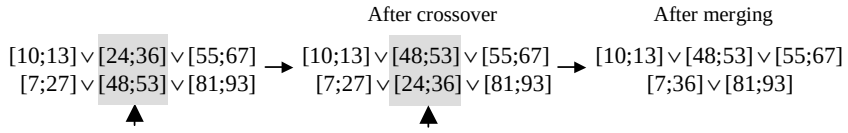


Figure 3.5. Interval merging after crossover operation. With a real-image, the crossover consists to exchange the tests corresponding to given parts of the spectrum of a sample. Interval merging does not change the mathematical interpretation of the test.

However, merging not only decreases the number of intervals in the rules, but also generates some information loss. In fact, in order to avoid a premature convergence of rules, it is generally important to preserve for the following generation two distinct intervals instead of a single aggregated one. 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. By checking the quality of the obtained rules, we finally discovered that *not merging* is a better solution if the user is not concerned by the computing time because it requires more iterations.

Mutation of classifiers. The mutation operator plays a dual role in the system: it provides and maintains diversity in a population of classifiers, and it can work as a search operator in its own right. The mutation processes a single classification rule and it creates another rule with altered condition structure or variables. The mutation operator may be applied on three levels: band level, interval level and border level.

Band mutation consists of a deletion of spectral bandwidth in a chosen classification rule. Its interest is twofold; firstly, the band mutation type allows for simplification and generalization of a rule; secondly, it allows for the elimination of noisy bands that frequently appear in hyper spectral images. The existence of noisy bands significantly perturbs the learning process, as well as the process of evolution convergence.

Interval mutation allows for a chosen band to add, eliminate or cut an interval in two spectral ranges. In case of addition, the new rule is completed by a new interval 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. And, this allows for the definition of a spectral tube in which spectral values of the pixels can be assigned to a given class.

Finally, border mutation modifies both boundaries of an interval. This mutation refines the idea of targeting spectral tubes carried out by the other types of mutation. It is worthwhile to note that the mutated rules are systematically validated.

In our system, mutation operators are dynamically adapted by tunning (i.e. we test different parameters for the same training set and we elect the best choice by looking at a testing set). Adjustment is related to the probability of each mutation operator according to its current effectiveness.

Generational replacement. The generational replacement is an operation that determines which of the current classifiers in the population is to be replaced by newly evolved classifiers. According to the algorithm A1, the new generation of classifiers is created from a population of parents (P_0) and their children after the crossover and the mutation operations (P_2). In our system, the following replacement strategies are applied:

- the revolutionary strategy in which only the population of the children completely replaces the parent population (P_0),
- the steady-state strategy in which new classifiers are inserted in the new population by replacing the worst, oldest member, or the most similar members, or by preserving the best classifiers (elitism).

There exist other replacement strategies integrating for instance the heuristics where the best individual of the previous population replaces the worst one of the current population or the heuristics where the new individuals having a performance higher than a certain threshold are inserted. However, both of these strategies present the risk of having individuals remain in the population. This is not necessarily a problem except in the case of a weak genetic pool in which some individuals of average performances that would profit from immunity.

3.5. From the rule creation to the evolution with XCS-R

XCS and XCS-R are learning classifier systems (LCS). LCSs are algorithms for symbolic learning. The main difference between LCSs and other learning techniques like neural networks and genetic programming is the aptitude for collaborative learning. The objective is to create a chain containing a given number of classifiers, so that the whole chain will be the complete solution of the problem. The information will be transmitted from a link to an other one, using messages to interpret.

XCS evolves a set of rules, the so-called *population of classifiers*. This method is presented for comparison with ICU. Rules are evolved by the means of a genetic algorithm. 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 [33], 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 (XCS-R) introduced in [36]. Recently, several studies have been reported that show that XCS performs comparably well to several other typical classification algorithms in many standard data mining problems [5, 4, 13].

This section provides a short introduction to the XCS classifier system. For a more detailed introduction to XCS and XCS-R the interested reader is referred to the original paper [33], the real-valued extension [36] and the algorithmic description [9].

3.5.1. 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[2]. 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.

[2]ICU benefits from an initialization creating directly good initial individuals by looking at the whole training set. However, XCS has this covering mechanism that could create good individual but by looking only at one sample of data. It is difficult to state which technique is better, because this strongly depends on the implicit patterns contained in the data. The difference of the accuracies of the two algorithms observed in the case study is caused by the preprocessing of the data (if applied) and the initialization phase. More theoretical studies are required to predict, in the general case, which method is more suitable.

3.5.2. 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 genetic algorithm (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 [8]. 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 [34] 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 [34] 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.

3.5.3. 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 (one pixel always corresponds to one class, contrary to the *one-to-N* classification, in which one pixel may correspond to several classes). 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:

$$S_r = \frac{\sum P_r * F_r}{\sum F_r} \tag{3.5}$$

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 have been 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. Here, contextual classification (i.e. looking at the neighbor pixels) can help to overcome this kind of ambiguity. Many cases of ambiguities occur when two classes have the same spectra whereas they can be discriminated by the context (for instance, water and shadows have relatively low reflectance and they can be distinguished by looking at the classes located around).

3.6. Case studies

3.6.1. 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, [3]). During this project, data was obtained from satellite or aircraft, at different scales and resolutions, providing a *multi-level view* of the ground [29]. 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 [31]:

- 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 georeferencing;
- 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. The first, 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. The second 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 chapter, a remote sensing image of San Felice (Lagoon of Venice) has been chosen (figure 3.6). This image contains multispectral data (CASI 15 bands), with 142x99 pixels, 16 bits per pixel and 1.3m terrain resolution [24]. 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.

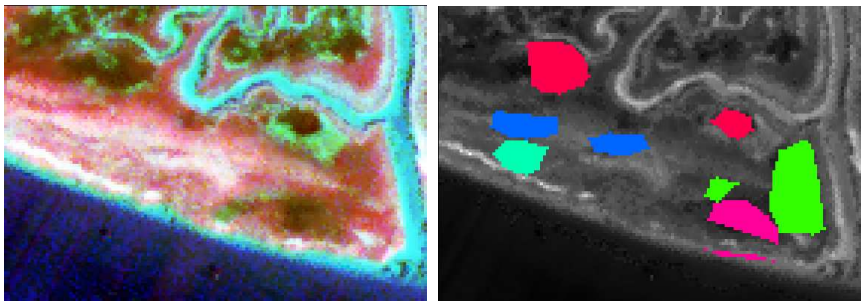


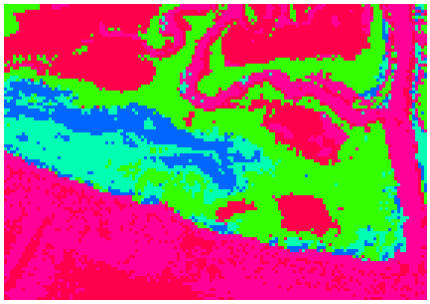
Figure 3.6. Reference and expert data (CASI, San Felice)

3.6.2. ICU classification

The image and statistics presented in the figure 3.7 summarize our experiments.

The parameters (described in the section 3.4.3) for ICU are as follows: P_{cross} = 0.7, P_{mut} = 0.15, $P_{mut,band}$ = 0.3, $P_{mut,interval}$ = 0.2, $P_{mut,border}$ = 0.4, 300 individuals, 2000 to 5000 generations and selection by rank for the crossover and the mutation operators.

It should be mentioned that validation points do not concern any water. Water can be classified quickly using standard statistical tools, even with unsupervised ones, and are beyond the interest of the experts. ICU classifies these pixels in the most approximate class (SPA1), but quality of these areas are not relevant (at the south and the right border). SPA2 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 spot at the right lower corner (SPA1 instead of SPA2, see the figure 3.7). The very narrow resemblance of the spectra of these



		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%	

Figure 3.7. Classified image and confusion matrix for ICU

two classes explains that. Classification is quite globally correct (κ-index of 0.81, average accuracy of 81.1%). The figure 3.8 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 (in fact in the color "0", depending on your printout). The map of overlaps is automatically generated, it shows pixel classes and the degree of mixing, without any external knowledge. The image below demonstrates, for instance, that no knowledge about water evolved.

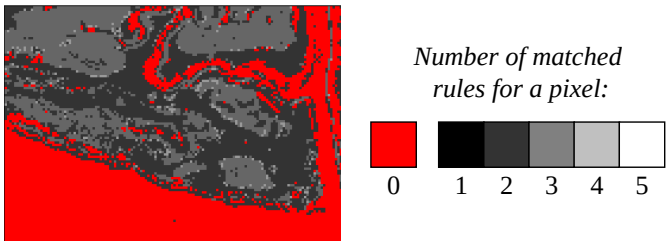
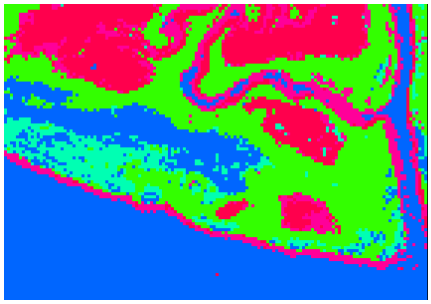


Figure 3.8. Map of the overlapping zones, produced only by ICU

3.6.3. XCS classification

The classification with XCS shows quite good results. The image and statistics presented in the figure 3.9 illustrate our experiments.

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 (a composition of several species on the same pixel). 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 (κ-index of 0.88, average accuracy of 87.7%). XCS is then better than ICU with a less accurate initialization procedure, because the pattern of this data could be better learned with local initializations (covering operator) than using a global initialization (*GenMinMax* or *GenSpectro* of ICU). But global initialization is more fast.



		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%	

Figure 3.9. Classified image and confusion matrix for XCS

The typical learning time with ICU on this dataset is about 2-3 minutes, against 4-7 minutes on a 3 GHz CPU.

3.6.4. Comparison between ICU and statistical classifiers

To attest that performance of ICU is comparable to other algorithms known to be really efficient in this domain [7, 16, 21, 22], this algorithm was tested on two data sets. These data correspond to two images of San Felice (Lagoon of Venice), with different spectral and spatial resolutions, namely CASI sensor (date: 2002, 15 bands, 754x293 pixels, resolution 1.3 m^2, 6 classes) and the hyperspectral MIVIS sensor (date: 2003, 20 bands, 396x170 pixels, resolution 2.6 m^2, 7 classes).

The results for the testing set (50% of all of the data) are presented in the table 3.2. Accuracy is the mean of the diagonal of the confusion matrix and the κ-index characterizes a correctly distributed confusion matrix (1 on the diagonal, 0 everywhere else).

	Accuracy for CASI (κ-index)	Accuracy for MIVIS (κ-index)
ICU	0.990 (0.99)	0.884 (0.88)
SVM	0.974 (0.97)	0.978 (0.98)
Neural network	0.893 (0.89)	0.878 (0.88)

Table 3.2. Results obtained for the two images.

For ICU, the parameters was the same that those described in the section 3.6.2. For the algorithm based on neural networks, a learning rate of 0.1, 100000 iterations, 1 hidden layer of 7 to 15 neurons, an incremental method for the learning, and a symmetrical sigmoid activation function were chosen. The exit layers represent the expert continuous values and there are as many exit neurons as there are classes to be learned. The free library in *C*, named the Fast Artificial Neural Network Library (FANN, [1]), was used. The chosen neural network topology was simple but efficient. For the algorithm based on Support Vector Machine (SVM), the RBF kernel was used; and the parameters C and γ were discovered for each

data set using free software and a step by step optimization algorithm in *Python* presented on the site of the LIBSVM [10].

ICU has shown better performance than SVM and NN for the CASI image and better than NN for the MIVIS image. The image of the MIVIS sensor was most difficult to analyze for several reasons: some additional bands; less pixels in the image, thus less samples in the training set (each time the whole training set consisted of less than 2% of the image); the first bands of MIVIS are slightly more disturbed than for CASI; and finally some classes of vegetation were no longer present on the saltmarsh in 2004, which caused a drop in the quality of the training data. When the values in the data set are a kind of composition of several pure classes (*mixels*) a system based on disjunctions like ICU proves to be more useful. Moreover, this kind of classifier would make it possible to expose the contents of the rules to a human expert contrary, for example, to a neural network (often treated as a *black box*).

To illustrate the simplicity of the formalism of the rules discovered by our algorithm, an example of a rule which classifies the instances of one of the class follows:

$(435 \leq B_0 \leq 1647) \wedge \ldots \wedge ((365 \leq B_{74} \leq 4023) \vee (15643 \leq B_{74} \leq 48409))$ $\wedge \ldots \wedge (668 \leq B_{79} \leq 4413) \Rightarrow$ [CLASS 4]

where B_i is the reflectance value for the band i of the considered pixel. The typical learning time with NN on this dataset is about 1 minute, against 10-15 minutes for the step by step optimization of SVM on a 3 GHz CPU.

3.6.5. Summary of experiments

The three case studies have demonstrated the high capacity of the evolution-based classifiers to interpret and classify heterogeneous and complex images (e.g. high dimension in (X, Y, S), large number of bands and noisy data that generate a computational complexity of $O(n^3)$). The quality of classification has been shown very high even in cases of high number of noisy bands and mixed pixels. 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 have been in general simple and easy to interpret by remote sensing experts. They are also mutually exclusive and maximally specific. Nevertheless, the learning time was relatively long due to the large image size and the chosen parameters. Classified images by the discovered rules have shown that the evolution-based classifier is able to faithfully reproduce the human expertise and algorithms already in use in this domain.

3.7. Conclusion and perspectives

This chapter has detailed the evolution-based classifier systems applied to remote sensing images. The systems have been able to discover a set of « *if* ... *then* ... *class* » classification rules using the fitness function based on image classification quality. These rules, which have been proven robust and simple to understand

by the user. The accuracy of the expert has been improved and the rules have been demonstrated sufficiently generic for reusing them on other parts of satellite images. Hence, the classifier systems can be considered of great interest when compared to traditional methods of classification.

Taking into consideration image complexity and noisy data, the results of our experiments are very encouraging. Case studies have demonstrated that the obtained classifiers are able to reproduce faithfully the terrain reality. The rules are well adapted to recognize large objects on the image (e.g. sport lands), as well as the smaller ones (e.g. trees, shadows, edges of the buildings). The redundant or noisy bands have been successfully identified by the classifiers. The formalism of rule representation has allowed the modelling of a spectral tube adapted to the granularity of spectral reflectance.

The potential of evolution-based classifiers in remote sensing image classification begins to be explored. Further investigation of the classifiers efficiency are necessary. Currently, we are starting to work on a more powerful representation of rules including spatial knowledge, temporal relations, and hierarchical representation of objects. Contextual classification can help to overcome some ambiguities, as stated in the last section before the case studies, but this technique requires to know in which order the algorithm will classify the neighbor pixels. Future research would be followed in this direction. We are also trying to optimize system performance, in particular the implementation of the genetic process on a parallel machine and the tuning of its initial parameters. The classifier system developed by this research work, called ICU, and other classification software related to remote sensing are currently available on our web site http://lsiit.u-strasbg.fr/afd. The XCS algorithm can be downloaded from the IlliGAL web site, http://www-illigal.ge.uiuc.edu.

Acknowledgements

The authors would like to thank Marco Marani, Enrica Belluco, Monica Camuffo and Sergio Ferrari from the UNPADU University (Padova, Italia) for the multispectral data sets used (images of San Felice and continuous expertise acquired during the TIDE project [3]). The authors are grateful to Mindy Jacobson and Pierre Gançarski, Cédric Wemmert, Christiane Weber, Anne Puissant and Massimo Menenti from the Louis Pasteur University (Strasbourg, France) for their advices, suggestions and comments during the early stages of this research, and also to David E. Goldberg and Martin Butz from the IlliGal Laboratory at Urbana-Champaign for providing hints, algorithms and judicious remarks concerning the XCS algorithm. We also thank our two reviewers for dedicating the time and expertise to improve the presentation and the readability of the chapter.

Bibliography

[1] Fast artificial neural network library, software available at http://fann.sourceforge.net/.

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

[3] 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.

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

[5] E. Bernadò, X. Llorà, and J. M. Garrel. Xcs and gale: A comparative study of two learning classifier systems and sic other learning algorithms on classification tasks. *Advances in Learning Classifier Systems (LNAI 2321), pages 115-132*, 2002.

[6] T. Blickle and L. Thiele. A comparison of selection schemes used in genetic algorithms. *Computer Engineering and Communication Networks Lab, TIK-Report Nr. 11, Second Edition, Swiss Federal Institute of Technology, Zurich*, 1995.

[7] M. Brown, H. G. Lewis, and S. R. Gunn. Support vector machines for optimal classification and spectral unmixing. *Ecological Modelling, 120, pages 167-179*, 1999.

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

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

[10] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm, 2001.

[11] J. M. Daida, D. E. Lund, C. Wolf, G. A. Meadows, K. Schroeder, J. F. Vesecky, D. R. Lyzenga, and R. Bertram. Measuring topography of small-scale waves. In T. Stein, editor, *Proceedings of IEEE International Geoscience and Remote Sensing*, pages 1881–1883, Florence, Italy, 1995. IEEE Press.

[12] K. A. DeJong. Learning with genetic algorithms: An overview. *Machine Learning, vol. 3, pages 121-138*, 1988.

[13] P. W. Dixon, D. W. Corne, and M. J. Oates. A preliminary investigation of modified xcs as a generic data mining tool. *Advances in Learning Classifier Systems (LNAI 2321), pages 133-150*, 2002.

[14] R. Fjørtoft, P. Marthon, A. Lopes, F. Sery, D. Ducrot-Gambart, and E. Cubero-Castan. Region-based enhancement and analysis of sar images. *Proceedings of ICIP'96, vol. 3, Lausanne, pages 879-882*, 1996.

[15] C. Fonlupt and D. Robilliard. Genetic programming with dynamic fitness for a remote sensing application. *Proceedings of Parallel Problem Solving from Nature (PPSN'2000), Paris*, 2000.

[16] X. Jiang, M.-S. Chen, M. T. Manry, M. S. Dawson, and A. K. Fung. Analysis and optimization of neural networks for remote sensing. *Remote Sensing Reviews, vol. 9, pages 97-114*, 1994.

[17] J. Korczak and N. Louis. Synthesis of conceptual hierarchies applied to remote sensing. *Proceedings of SPIE, Image and Signal Processing for Remote Sensing IV, Barcelona, pages 397-406*, 1999.

[18] J. Korczak and A. Quirin. 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, April 2003*, 2003.

[19] T. Kurita and N. Otsu. Texture classification by higher order local autocorrelation features. *Proceedings of Asian Conf. on Computer Vision, Osaka, pages 175-178*, 1993.

[20] P. L. Lanzi. A study of the generalization capabilities of xcs. *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97), Morgan Kaufmann*, 1997.

[21] X. Liu, K. Chen, and D. Wang. Extraction of hydrographic regions from remote sensing images using an oscillator network with weight adaptation. *Ieee Transactions On Geoscience And Remote Sensing, Vol. 39, No. 1*, 2001.

[22] M. T. Manry, S. J. Apollo, L. S. Allen, W. D. Lyle, W. Gong, M.S. Dawson, and A. K. Fung. Fast training of neural networks for remote sensing. *Remote Sensing Reviews, vol. 9, pages 77-96*, 1994.

[23] E. Ozcan and C. K. Mohan. Partial shape matching using genetic algorithms. *Pattern Recognition Letters, 18, pages 987–992*, 1997.
[24] A. Quirin. Discovery of classification rules : evolutionary classifiers. *MSc Thesis, Louis Pasteur University, LSIIT/CNRS, Strasbourg*, 2002.
[25] M. V. Rendon. Reinforcement learning in the fuzzy classifier system. *Reporte de Investigaci No. CIA-RI-031, ITESM, Campus Monterrey, Centro de Inteligencia Artificial*, 1997.
[26] R. A. Richards. Zeroth-order shape optimization utilizing a learning classifier system. *Book published online, http://www.stanford.edu/ buc/SPHINcsX/book.html, Stanford*, 1995.
[27] R. L. Riolo. Empirical studies of default hierarchies and sequences of rules in learning classifier systems. *PhD Dissertation, Comp. Sc. and Eng. Dept, Univ. of Michigan*, 1988.
[28] B. J. Ross, A. G. Gualtieri, F. Fueten, and P. Budkewitsch. Hyperspectral image analysis using genetic programming. *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pages 1196-1203.*, 2002.
[29] D. A. Stow, A. Hope, D. McGuire, D. Verbyla, J. Gamon, F. Huemmrich, S. Houston, C. Racine, M. Sturm, K. Tape, L. Hinzman, K. Yoshikawa, C. Tweedie, B. Noyle, C. Silapaswan, D. Douglas, B. Griffith, G. Jia, H. Epstein, D. Walker, S. Daeschner, A. Petersen, L. Zhou, and R. Myneni. Remote sensing of vegetation and land-cover change in arctic tundra ecosystems. *Remote Sensing Environment, 89, pages 281-308*, 2004.
[30] T. Toutin and P. Cheng. Quickbird - a milestone for highresolution mapping. *Earth Observation Magazine, 11(4), pages 14-18*, 2002.
[31] D. Verbyla and T. Hammond. How to lie with an error matrix. *Remote Sensing & Image analysis (online paper avaiable on http://nrm.salrm.uaf.edu/ďverbyla/online/errormatrix.html)*, 2002.
[32] C. Weber. Images satellitaires et milieu urbain. *Hermès, Paris*, 1995.
[33] S. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation, 3(2), pages 149-175*, 1995.
[34] S. W. Wilson. Generalization in the xcs classifier system. *Proceedings of the Third Annual Genetic Programming Conference, pages 665-674*, 1998.
[35] S. W. Wilson. State of xcs classifier system research. *Proceedings of IWLCS-99, Orlando*, 1999.
[36] S. W. Wilson. Get real! xcs with continuous-valued inputs. *Learning Classifier Systems: From Foundations to Applications, LNAI 1813, pages 209–220*, 2000.

ARNAUD QUIRIN: UNIVERSITÉ LOUIS PASTEUR, LSIIT/CNRS, STRASBOURG, [redacted] FRANCE, QUIRIN@LSIIT.U-STRASBG.FR

JERZY KORCZAK: UNIVERSITÉ LOUIS PASTEUR, LSIIT/CNRS, STRASBOURG, [redacted] FRANCE, KORCZAK@DPT-INFO.U-STRASBG.FR