EVOR-STACK: A label-dependent evolutive stacking on remote sensing data fusion

Jorge García-Gutiérrez , Daniel Mateos-García, José C. Riquelme-Santos

Keywords:

Data fusion
Ensembles
Evolutionary computation
Feature weighting
Label dependence
Remote sensing
Hybrid artificial intelligence systems

A B S T R A C T

Land use and land covers (LULC) maps are remote sensing products that are used to classify areas into different landscapes. Data fusion for remote sensing is becoming an important tool to improve classical approaches. In addition, artificial intelligence techniques such as machine learning or evolutive computation are often applied to improve the final LULC classification. In this paper, a hybrid artificial intelligence method based on an ensemble of multiple classifiers to improve LULC map accuracy is shown. The method works in two processing levels: first, an evolutionary algorithm (EA) for label-dependent feature weighting transforms the feature space by assigning different weights to every attribute depending on the class. Then a statistical raster from LIDAR and image data fusion is built following a pixel-oriented and feature-based strategy that uses a support vector machine (SVM) and a weighted k-NN restricted stacking. A classical SVM, the original restricted stacking (R-STACK) and the current improved method (EVOR-STACK) are compared. The results show that the evolutive approach obtains the best results in the context of the real data from a riparian area in southern Spain.

1. Introduction

Remote sensing is an important discipline for many tasks such as resource management [1], environmental monitoring [2] and disas-ter response [3]. For a long time, machine learning techniques have been used to improve remote sensing performance and applicability. In addition, the use of active sensors such as LIDAR (light detection and ranging) has recently spread to improve the classical remote sensing products [4], which were mainly based on images. This change involves a data complexity increase and makes artificial intelligence systems and data fusion techniques even more impor-tant for extracting meaningful information from remote sensing data.

Remote sensing knowledge can be gathered in several products, among which land use and land covers (LULC) maps are arguably one of the most important. LULC maps are based on a classification of the terrain depending on its morphologic or functional character-istics, and they are a very remarkable tool in the development of policies to manage the natural environment. Automatic pixel classi-fication, which is generally supervised, is usually the first step to extract maps from remote sensing data. Several techniques from machine learning have been used in this context with satisfactory results, e.g., k-NN [5], Naive Bayes [6] and SVM [7].

Although the validity of machine learning has been widely demonstrated in the remote sensing context, more research is needed to fulfil the standard requirements of many remote sensing products, and especially for LULC maps [8]. Thus, the final classification has to maintain not only the global accuracy that is the general standard but also satisfactory partial accuracies for every label. Thus, some researchers [9,10] have started to exploit hybrid artificial intelligence systems [11] based on opti-mization techniques (genetic algorithms) and classical machine learning applied to remote sensing data.

Evolutionary computation is usually used to search optimal weighting for both structural and functional aspects to improve the predictive models for machine learning. In supervised machine learning, there are essentially three main areas of weighting application: support vector machine optimization, artificial neural networks (training and topology) and feature weighting.

Support vector machines (SVMs) are learning algorithms proposed by Vapnik [12,13]. A SVM constructs one or more hyperplanes in a high-dimensional space by means of a kernel function. Therefore, the kernel function election and its proper parametrization are critical for the performance of the classifier. Many authors have used evolutionary computation to solve this problem with pure [14] or real-coded [15] genetic algorithms. Other authors have also explored the use of genetic programming for kernel assembling [16] or developed hybrid algorithms [17], which usually have an evolutionary module in a first level and a SVM applied for classification in a second one.

Artificial neural networks (ANNs) consist of a simulation of the structures and behaviour of biological neural systems by means of mathematical models [18]. Evolutionary computation has been used to train the set of neural network parameters and to design its structure. From the viewpoint of training the network, the common

approach is to create the genome by encoding the weights of the connections. This may be done by typical bit-based encoding, but there are also more efficient proposals [19]. The main problem with the approaches based on genetic algorithms is the lack of efficient crossover operators because it is difficult to establish which functional parts of the network are to be exchanged. For this reason, other techniques based on genetic programming have been more successful [20]. There have also been several studies on evolutionary computation applied to the design of neural network architecture and weighting optimization. In these cases, the fitness function is usually multi-objective [21] because it must take into account different aspects (structural and functional) of the network.

Techniques that use genetic algorithms to find a set of weights for the feature space, allowing greater accuracy in the classification process, are common in the literature [22]. The usual individual encoding is a set of real values that represent the weights of each feature. The fitness is defined by the classification process itself. Therefore, the search process can be viewed as a global task in which the optimal weights are considered in terms of their features regardless of the label assigned to each instance. Moreover, the use of several evolutionary techniques (genetic algorithms and evolutionary strategies) for both instance selection and feature weighting has proven possible [23], and an optimal weight searching dependent on each label has recently been tested [24] with good results in biomedicine.

With all this in mind, this work can be seen as a new application of hybrid artificial intelligence system [25] which combined the application of ensembles in remote sensing [26,27] that takes advantage of contextual information from multi-source (LIDAR and aerial images) data and the use of evolutive computation to improve the separability of pixels for each label. Thus, we improve a method called R-STACK [28] (based on the stacking of a SVM and multiple k-NN classifiers) with a matrix of weights obtained in the preprocessing stage [29] to give rise to a new method called EVOR-STACK for the following three purposes:

- Improve the general accuracy of an automatically generated LULC map.
- Show the quality of models when hybrid artificial intelligence systems are applied to LIDAR and imagery fusion data.
- Obtain information about what features are the most important to classify each landscape by studying the resulting weights per label.

The rest of this paper is organized as follows: Section 2 presents the study area for this work and provides a brief description of the different landscapes in the area. Section 3 provides a detailed description of the proposed method. The results and discussion are presented in Sections 4 and 5, respectively. Finally, Section 6 is devoted to summarizing the conclusions and to discussing future lines of work.

2. Data description

A LIDAR system is a remote sensor technology that is able to register object heights. The process starts with the emission of light (usually laser). The light impacts on a surface and its reflected signal is caught by the LIDAR system. Finally, the system measures the time elapsed from emission to reception to establish the distance between the emitter and the object that produced the return. This process gives rise to a cloud point database in which for every point, it is possible to obtain the following data: spatial position (i.e., x, y and z coordinates), intensity of return and number of returns in a sequence (if a pulse caused multiple impacts). These measurements and the RGB values in an orthophoto are used in this work to obtain statistical features on which the whole classification is based.

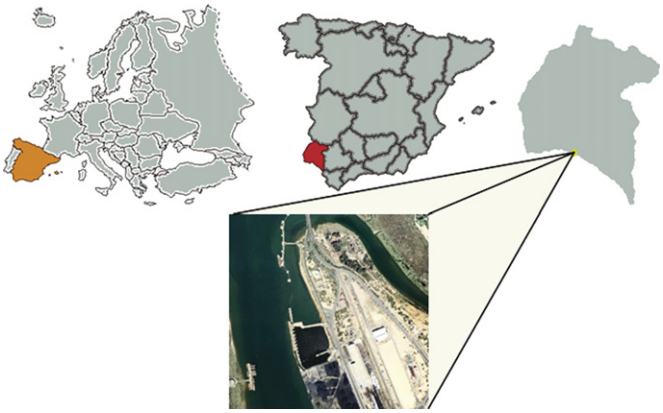

Fig. 1. Study area.

Our LIDAR data were collected in coastal areas of the province of Huelva (Fig. 1). The pulses were geo-referenced and correctly validated by the distributor of the data and included 1,384,875 records for an area of 1.5 km^2. The reported precision indicates a maximum error of 0.5 m in the x–y positions and 0.15 m in the z position. Along with the LIDAR flight, aerial photographs were taken of the area with a resolution of 0.5 m^2. The study area is situated in southern Spain at the mouth of the Tinto and Odiel rivers. This area is near the city of Huelva and presents a mix of urban and natural areas. The natural areas can be classified into five subclasses: watered zones, marshland and vegetation (low, middle and high). The high vegetation in the area consists of scarce trees of the genus *eucalyptus*. The middle vegetation consists of different types of Mediterranean bushes that principally surround roads and urban areas. Pastures are classified as low vegetation and include bare earth areas. The urban areas are also classified into three subclasses: roads and railways, dumps and urban areas (buildings and industrial areas).

3. Method

The method proposed, called EVOR-STACK (steps 4–8 in Algorithm 1), is a new contextual [30] hybrid method to improve thematic maps by means of a remote sensing data fusion, evolutionary computation and complex classifiers (ensembles) [31].

Algorithm 1. LULC classification method.

input
l: LIDAR data
o: Orthophotograhy data
output
m: LULC map
begin
1. Build a matrix *raster* in which every cell involves a physical position with the corresponding statistics from *l* and *o*
2. Select a training set from *raster*, called *train*
3. Label each pixel in *train* using expert knowledge
4. Execute a multi-label EA to extract the matrix *W*
5. Let *svm* be a SVM model from *train*
6. Use *svm* to classify every pixel in *raster*
7. For each pixel *p* in *raster*
   7.1. Collect the neighborhood of *p* in a set *s*
   7.2. Use *W* to modify every pixel from *s*
   7.3. Build a weighted-distance k-NN model, *knn*, from *s*
   7.4. Use *knn* to classify *p*
8. Return a map *m* with every pixel spatial position and its label
end

The first step is the generation of a raster with a set of statistics to obtain a feature-based data fusion representation. It is important to set up a resolution according to each data source. For our study area, we work with a 3 m^2 resolution. To extract the object heights, a digital elevation model is needed to construct the real heights from the coordinate *z*. For our area, the method described in Gonçalves et al. [33] is selected.

The second step is done by an EA (evolutionary algorithm), which is used to obtain a multi-label weighting matrix [29]. This matrix provides an optimized set of weights to improve the final classification, as will be seen later.

Finally, an R-STACK [28] method is applied to obtain the final map. The set of weights from the previous phase is used to modify the feature space on the second level of the R-STACK method. In this way, a more accurate separation among neighbours is possible. In the following subsections, a detailed description is presented for every step.

3.1. Feature extraction and pre-process

The process presented in this article is a feature-based approach that fuses information from aerial images and LIDAR to generate high quality and detailed thematic maps. In this way, the first step is to calculate a set of variables from the image RGB values, LIDAR intensity, heights and their distributions for each pixel. Thus, 70 different features are calculated for every pixel which are mostly extracted from the literature [34,35]. In Table 1, a summary of these features can be seen. To the best of our knowledge, the use of some of them for data fusion is original work, e.g., the number of empty neighbours (NEMP) or the features based on the simulated normalized difference vegetation index (SNDVI). The NEMP feature is extracted from LIDAR and represents the absence of information, which is useful to detect watery areas because LIDAR is not able to reflect off of water. The SNDVI has proven useful for simulation of the classical normalized difference vegetation index (NDVI). The NDVI value is generated from the near infrared band (NIR) and the red band (R), as can be seen in Eq. (1). In our case, it cannot be calculated because the NIR band is not available in LIDAR or orthophotography. Thus, the new band SNDVI is used to simulate the NDVI using the intensity (I) from LIDAR (Eq. (2)) as a near-infrared value that approximates the real NIR value.

$$NDVI = \frac{NIR - R}{NIR + R} \tag{1}$$

$$SNDVI = \frac{I - R}{I + R} \tag{2}$$

Before generation of the model, a pre-process has to be carried out. Three different filters are executed. First, every missing attribute value is replaced with the corresponding average value. Then the data are normalized. Finally, a Correlation Feature Selection method (CFS from Weka [36]) with default parameters is applied to reduce the search space (see variables in bold in Table 1). With the selected features already generated, the next phase is the execution of the EA, which is characterized in the next subsection.

Table 1
Candidate variables. In bold, the final selected features. Variables with ∗ are calculated for each band of a pixel: height (H), intensity (I), red (R), green (G), blue (B) and simulated NDVI (SNDVI).

Variable	Description
CRR	Canopy relief ratio
CV ∗	Coefficient of variation
MIN ∗	Minimum
MAX ∗	Maximum
STD ∗	Standard deviation
AVG ∗	Average
NEMP	Number empty neighbours
VAR ∗	Variance
SKEW ∗	Skewness
KURT ∗	Kurtosis
RANGE ∗	Range
NOTFIRST	Second or later return
INTRASLP	Intra-pixel slope
EXTRASLP	Inter-pixel slope
PEC	Penetration coefficient
TOTALR	Total of returns
PCTN1	Unique return percentage
PCTN2	Double return percentage
PCTN3	Three or more returns percentage
PCTR1	First return percentage
PCTR2	Second return percentage
PCTR3	Third or later return percentage
PCTR31	PCTR3 over PCTR1
PCTR21	PCTR2 over PCTR1
PCTR32	PCTR3 over PCTR2

3.2. Evolutionary weighting algorithm

The basic structure of an EA can be seen in Algorithm 2: first, a random initial population of solutions is built, and the individual fitness of each solution is evaluated. Then each generation is formed from the previous one by crossing and mutation. Thus, the best solution is determined step by step through natural selection.

Algorithm 2. Evolutionary algorithm.

```
Build the initial population of individuals
Evaluate the fitness of each individual and save the best individual
while not termination do
    Select several individuals for reproduction according to a criterion
    Create new individuals through crossover and mutation operations
    Evaluate the fitness of new individuals and save the best individual
    Replace the population with the new individuals
end while
```

The goal of the proposed EA is to find an optimal matrix of real values to weight the features selected in the previous phases. Thus, the matrix has a row for each label and a column for each feature, and each cell contains a weight that is used to complete the classification process in three steps:

1. The weights are applied to the training instances according to their label.
2. Given a test instance, the weighting matrix is utilized to define a per-label weighted distance.
3. Then the test instance is classified by the nearest neighbour label calculated using the distance defined in the previous step.

A deeper description of the EA and its characteristics is provided in the next paragraphs.

3.2.1. Individual codification

To execute the evolutive algorithm, an individual description is required. In this case, an individual of the population is a matrix whose cells each represent a weight for a label and a feature. Hence, for a training set, there is a row for each label that has as many columns as features, so the initial population is a set of matrices of *b* rows and *f* columns, where *b* is the total number of labels and *f* is the total number of features. In addition, the initial population is built by initializing each cell of every matrix with a value randomly chosen from the interval $[-1, 1]$.

3.2.2. Fitness function

The training data consist of a matrix P with t rows (each representing a normalized feature instance, from now on a pixel) and f columns (one per feature). A class label is assigned with the label function on each instance of P. For simplicity, we assume that the label is an integer between 1 and b. Thus, a pixel p_i is a row of P (a vector of $[0, 1]^f$ such that $label(p_i) = l \in \{1..b\}$). A transformation is given by an individual $W = [w_{ij}]_{b \times f}$. Thus, a pixel p can be transformed to p^l by a label l according to the following equation:

$$\forall j = 1 \ldots f : p_j^l = w_{lj} * p_j \tag{3}$$

A particular case can be seen when the label of the instance to transform is known. In this case, we denote p' as the transformed pixel, and thus, p' is defined as

$$p' = p^{label(p)} \tag{4}$$

As seen in Algorithm 3, the training set P is divided into n bins (step 3). The weights of the individual that are being evaluated are applied to $n-1$ bins (step 5), obtaining the set P' by means of Eq. (3), and the remaining bag is used as the initial test (step 6 et seq.). The nearest pixel from P' to each pixel e from the test bin B_k is calculated (steps 6–9) according to the distance d_W defined in Eq. (5).

$$d_W(e,p') = d_{Euclidean}(e^{label(p')}, p') \tag{5}$$

Algorithm 3. Fitness function.

input
W: Weight matrix
P: Pixel matrix
label: a function that returns a pixel label for every pixel.
output
fitness: classification error which is the objective function to be minimized.
begin
1: fitness = 0
2: **for all** $i = 1$ to m **do**
3: We divide P into n bags: $B_1, \ldots, B_n$
4: **for all** bag B_k **do**
5: According to Eq. (4), we apply the W transformation to every pixel from the remaining $n-1$ bags, obtaining the set of pixels P'
6: **for all** pixel p_i in B_k **do**
7: **for all** label $l \in \{1..b\}$ **do**
8: We construct the transformed pixel p_i^l according to Eq. (3)
9: We calculate $d_l =$ minimum distance from p_i^l to the pixels of P' according to Eq. (5)
10: We apply the W transformation to p_i according to its nearest neighbour label, and we add it to P'
11: **end for**
12: We calculate the minimum from the distances d_l. Let $h \in \{1..b\}$, the label of the pixel of P' that gives rise to d_l.
13: **if** the original test label of $p_i \neq h$ **then**
14: $fitness = fitness + 1$
15: **end if**
16: **end for**
17: **end for**
18: **end for**
end

The nearest neighbour of each test pixel according to the distance defined in Eq. (5) is returned. If its assigned label does not match its original test label, its fitness is increased (steps 13–14). In addition, once a test pixel has been transformed with the nearest neighbour weights, it becomes part of P', reinforcing the training (step 10).

3.2.3. Crossover and mutation

In the design of an EA, it is always important to establish a coherent search criterion in the space of possible solutions, especially if the encoding of the individuals belongs to $\mathbb{R}$. This can only be achieved with a proper selection of crossover and mutation operators.

A crossover operation for two individuals selected by the roulette-wheel method is applied to every corresponding row (the ith row of an individual is crossed with the ith row of the other one) because they have the same label.

In addition, two techniques have been selected for the generation of the new individuals: the uniform crossover and the BLX-α crossover [37]. Both techniques are mutually exclusive and one of them is randomly chosen to generate each new individual.

The mutation operator has been defined to increase or decrease the value of a weight according to a probability p. The increase or decrease is a random value δ that satisfies

$$\delta = r/z$$

where

$$r \in [0,1]$$

chosen randomly and

$$z \in \mathbb{N}$$

In this case, z is a decreasing value selected empirically for the evolutive process so that the variation is higher in the first generations and lower in the latest ones.

3.3. R-STACK method

Once the weighting matrix is obtained (step 4 in Algorithm 1), the R-STACK method is applied. R-STACK is based on a modified stacking of two well-known classifiers (SVM and k-NN). To generate the SVM model, the SMO [38] Weka implementation is used. The second level of the R-STACK method is implemented by means of an ad hoc k-NN.

In this way, the stacking general scheme is modified to adapt it to geographic data. The classification task is then done in two steps: first, the SVM takes every non-weighted feature from the pixels in the training area to build an initial model that classifies every pixel from the study zone (steps 5 and 6 in Algorithm 1). At that point, a classical SVM application to the images is obtained. Later, a specific model is built for each pixel taking the feature values of its neighbours in the pixel raster as a training set, which involves a strong relationship (contextual dependence) among the training pixels and the current pixel (step 7 Algorithm 1). In the end, the k-NN classifies the current pixel using the model built by its weighted neighbours according to the distance described in Eq. (5). This step has been modified from the original R-STACK method.

The number of neighbours and the level of adjacency are selected empirically. For the study area, we work with $k = 3$ and 8-adjacency, i.e., each 3-NN is developed with just eight instances of the pixel surrounding area.

4. Results

To establish the accuracy of EVOR-STACK, it is compared with two other classifiers: classical SVM and R-STACK. This comparison is based on two well-known testing strategies: a hold-out process and a 10-fold cross-validation (10-FCV).

Table 2
Per-class results of the hold-out test for every method.

Class	SVM		R-STACK		EVOR-STACK	
	TP-Rate	F-Mea.	TP-Rate	F-Mea.	TP-Rate	F-Mea.
Water	0.999	0.988	1	0.991	0.999	0.998
Marsh	0.925	0.847	0.952	0.886	0.897	0.939
Roads	0.888	0.886	0.916	0.91	0.912	0.949
Low. veg.	0.834	0.854	0.851	0.868	0.894	0.942
Middle veg.	0.668	0.697	0.735	0.776	0.845	0.916
High veg.	0.711	0.69	0.781	0.74	0.851	0.917
Buildings	0.816	0.871	0.843	0.9	0.881	0.934
Dumps	0.679	0.807	0.818	0.898	0.976	0.988
Minimum	0.668	0.69	0.735	0.74	0.845	0.917
Average	0.815	0.83	0.862	0.871	0.907	0.948
Global average Accuracy (%)	88.1		90.8		92.21	

Table 3
Per-class results of the 10-FCV for every method.

Class	SVM		R-STACK		EVOR-STACK	
	TP-Rate	F-Mea.	TP-Rate	F-Mea.	TP-Rate	F-Mea.
Water	0.997	0.986	0.994	0.985	0.994	0.993
Marsh	0.939	0.902	0.967	0.932	0.992	0.983
Roads	0.909	0.906	0.928	0.909	0.978	0.969
Low. veg.	0.958	0.906	0.966	0.919	0.984	0.985
Middle veg.	0.77	0.767	0.792	0.827	0.948	0.966
High veg.	0.882	0.883	0.915	0.912	0.98	0.969
Buildings	0.861	0.914	0.887	0.934	0.979	0.985
Dumps	0.476	0.611	0.564	0.706	0.92	0.954
Minimum	0.476	0.611	0.564	0.706	0.92	0.954
Average	0.738	0.799	0.779	0.846	0.972	0.976
Global average Accuracy (%)	91.38		93.09		98.23	

A hold-out process is the traditional testing in remote sensing. Thus, every algorithm is trained with 618 instances (training set) and the accuracies are checked with a test set (7501 non-training instances) with both sets extracted by visual inspection. Due to its random origin, multiple execution for an EA is recommended to establish its quality [39]. In our case, the results shown for EVOR-STACK are the average case obtained over three evolutive processes.

Table 2 shows the per-class detailed accuracies for every method. Concretely, it shows each class value for the true positive rate (TP-Rate) and the harmonic mean of the precision and recall (F-measure), which is described in Eq. (8). The F-measure is based on the values of TP, FP and FN, which are the total positives, false positives and false negatives, respectively, calculated from the confusion matrix. It also provides the per-class average and minimum across the different partial results and the global accuracy attained by each algorithm. It is important to underline that every SVM is obtained by the SMO algorithm with its default parameters in Weka so that the differences among EVOR-STACK, R-STACK and classical SVM are due to underlying structural characteristics and not to different parameterizations.

$$recall = \frac{TP}{TP+FN} \tag{6}$$

$$precision = \frac{TP}{TP+FP} \tag{7}$$

$$F\text{-}measure = \frac{2*precision*recall}{precision+recall} \tag{8}$$

The second type of testing is a 10-FCV. We select all the classified pixels as the data set (training and test sets, 8120 instances). The per-class and averaged accuracy results from the 10-FCV for every approach can be seen in Table 3, in which the average instance of EVOR-STACK obtains the best results.

A comparison based on algorithm ranks is also established to complete the comparison among the approaches. For this purpose, we need a set of results for every approach on several distinct data sets. Because remote sensing data are expensive to generate, the comparison has to rely on an artificial data split. In our case, 10 splits are created from the original test data so that each split contains about 812 instances. Then a 10-FCV process is made for every split. The results (100 partial values) are then registered. In this way, a fair comparison of the algorithms can be obtained by average ranks. According to the 100 registered partial results, our evolutive approach ranks first as can be seen in Table 4, which shows the average ranking for each classifier. In this case, a value of 1 for a rank means that a classifier is the best for a split, while a rank of 3 implies that it is the worst. Therefore, the ideal objective for our approach is to reach an average ranking value of 1, which would mean EVOR-STACK would be the best for every split.

Table 4
Average rankings of the algorithms.

Algorithm	Ranking
SVM	2.645
R-STACK	1.96
EVOR-STACK	1.395

A comparison alone is not enough to show the superiority of a given approach, as has been noted by several authors [40]. We have to demonstrate that the differences among the rankings of the algorithms are significant. Our purpose is to compare our approach with the other two classifiers in terms of accuracy. Since the results are not normally distributed (this condition is certified by the D'Agostino-Pearson test [41]), it is not possible to apply a parametric test like ANOVA. Therefore, we utilize a non-parametric procedure for robustly comparing classifiers across multiple data sets [42] to evaluate the statistical significance of the measured differences in algorithm ranks. The chosen procedure involves the use of the Friedman test and the Holm post hoc procedure (see [41] for a complete description of both non-parametric approaches).

The null hypothesis for the Friedman test is that the ranks are not significantly different (their averages are not sufficiently different from the mean rank $r=2$). When the test is applied to assure the significance level of the results, its p-value is lower than $7.27E-11$, so the null hypothesis is rejected. The next step is the use of the Holm method. This procedure is specially indicated for rigorous comparisons to detect significant pair-wise differences among all the classifiers. Thus, for our study, the null hypothesis is that there exists a pair-wise comparison between an algorithm and our control method, EVOR-STACK, that does not show significant differences. The p-value obtained is lower than the required value for every pair-wise comparison (see column Holm in Table 5), so the null hypothesis is rejected. Having found that the measured average ranks are significantly different (at $\alpha = 0.05$), our analysis based on ranks reveals that the accuracy of EVOR-STACK is significantly better than that of the other approaches.

The label-dependent feature-weighting evolutive algorithm provides the weights for every feature according to each class. This information also permits us to select the best possible features to distinguish among classes. For the study area, the three features with the highest weights by class can be seen in Table 6.

Finally, the resulting thematic maps for every classifier can be seen in Fig. 2 beside the LIDAR intensity image used for this study,

the original orthophoto of the area and the official LULC map of the Regional Ministry of Andalusia.

5. Discussion

This study provides important facts that have to be taken into account. The first is related to global accuracy (see Table 2). For our study area, the three methods obtain overall accuracies over 85%, which suggests that a feature-based approach is very suitable for thematic map generation. Moreover, the average accuracy for the EVOR-STACK is 92.21%, which is 1.45% better than the original R-STACK, which obtains a 90.76% accuracy for the hold-out process. Both methods improve the results of the classical SVM, which provides near 88% accuracy. These differences are even greater if we focus on the 10-FCV results.

A deeper study of the results shows that the most problematic classes are middle and high vegetation. The differences among them can be slight in some cases (high bushes vs. low trees), and some intensity values can falsify the classification because multi-impact intensity can be affected by noise hard to deal with [32]. In this context, the intensity is modified and the actual value is unknown. To avoid this problem, some authors do not consider the multi-impact returns and instead delete them from the final study. Because we are working with low-resolution LIDAR data (0.5 pulses/m^2 approx.), such a solution is not possible for us. Thus, correction of the LIDAR intensity is still an important issue, and for our feature-based approach, the only possible way to solve this problem is the definition of vegetation indexes that are not so dependent on LIDAR intensity or the addition of new bands provided by other sensors (e.g., infrared cameras). Dumps have also been detected as another problematic class specially when we analyze the 10-FCV results. This class produces worse results for this test than for the hold-out process, which leads us to consider the problem of imbalance data. This problem will be discussed later and may also affect the case of high and middle vegetation in the same way, although its effects are less visible.

The insufficiency of a global accuracy of only 85% when attempting to generate a good thematic map is well-known [8]. Thus, every class has to be over the 85%, but this condition is barely satisfied even for official maps. In this context, another important conclusion extracted from our study is that EVOR-STACK is useful to fulfil this requirement. Looking at the true positive rate of each method, it is possible to recognize that only the EVOR-STACK method shows an accuracy above 84.5% for each class, whilst the other methods have classes with below 80% accuracy. Looking at the 10-FCV results, this finding is even clearer (Table 3).

It is also important to underline that although the tests performed obtain good results, the post-classification visual inspection of the resulted map is not completely satisfactory. Salt and pepper noise is still a problem that undermines the visual quality of the

Table 5
Results of the Holm procedure comparison when EVOR-STACK is compared with each other algorithm for $\alpha = 0.05$.

i	Algorithm	$z = \frac{R_0 - R_i}{SE}$	p	Holm
2	SVM	8.839	9.7E−19	0.025
1	R-STACK	3.995	6.5E−5	0.05

Table 6
Three most important features according to their weights for every label in the study zone.

Class	Features		
1	HRANGE	GKURT	INTRASLP
2	HMAX	INTRASLP	RMIN
3	IAVG	GKURT	RMIN
4	HAVG	RVAR	IAVG
5	INTRASLP	TOTALR	HMAX
6	HMAX	SNDVIAVG	INTRASLP
7	SNDVIAVG	GCV	SNDVIMAX
8	RVAR	INTRASLP	TOTALR

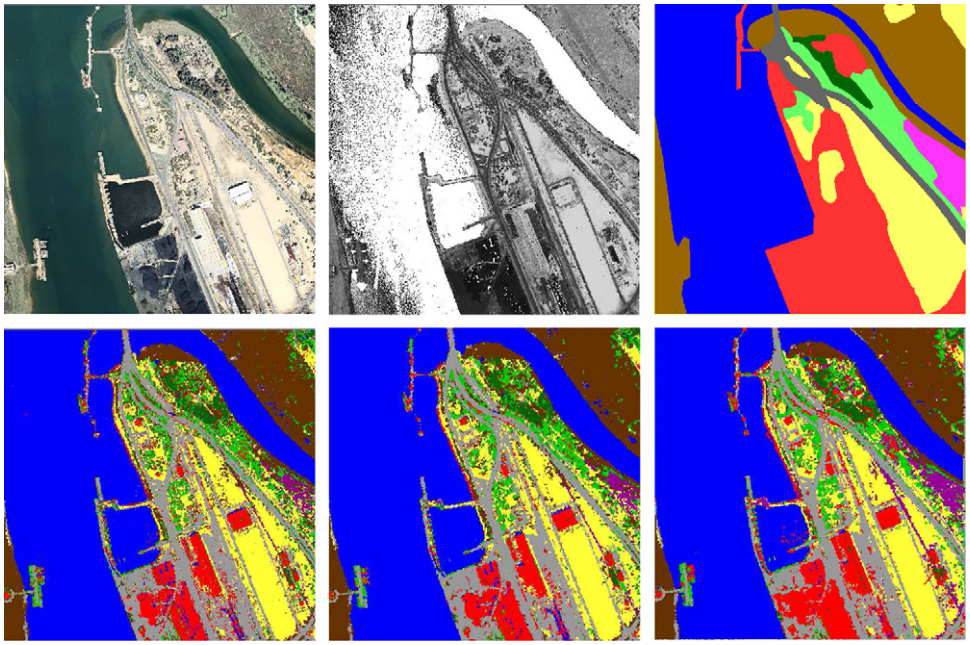

Fig. 2. First row: orthophoto, LIDAR intensity image and official LULC map. Second row: final classification obtained by SVM, R-STACK and EVOR-STACK, respectively. Colour legend: blue for water, grey for roads and railways, brown for marshland, yellow for low vegetation, lighter green for middle vegetation, green for high vegetation, purple for dumps and red for buildings and other human-impacted areas. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 7
Number of training instances for every class.

Class	Training size
Water	2167
Marsh	1246
Roads	1077
Low vegetation	682
Middle vegetation	484
High vegetation	403
Buildings	1239
Dumps	204

map. This behaviour is possibly caused by imbalance problems. The three problematic classes (dumps, middle and high vegetation) have a very low number of training instances compared with other classes like water or buildings (see Table 7). The imbalance problem has also been detected by other authors in the context of remote sensing [43–45] and some interesting approaches have recently appeared in the literature [46–48] which will have to be taken into account in future research.

Lastly, after the application of the EA, every class has its own set of features that best determine its label. This information provides an important feature selection tool and allows us to establish a more accurate class separation. The importance of a sensor can be evaluated easily by noting which features are more important for each class. In our case, LIDAR has the greatest significance level for almost every class, as expected. Table 2 contains three features per label with the highest coefficients. For our study area, INTRASLP and HMAX are the most frequent in the list because the classification relies mainly on heights and differences among neighbouring heights. The proportion between LIDAR and image variables is also important. In our case, it is almost the same (four image features, six LiDAR features and two mixed indexes). In addition, note that two of the most important selected features are derived from SNDVI, which demonstrates the importance of this band in the final classification.

6. Conclusions

In this paper, we presented a hybrid artificial intelligence method called EVOR-STACK based on a multiple-classifier ensemble to improve LULC map accuracy. The method worked at two processing levels. First, a label-dependent feature-weighting EA transformed the feature space, assigning different weights to every attribute depending on each class. Then the second level constructed a statistical raster from LIDAR and image fusion data following a pixel-oriented and feature-based strategy. Finally, the data were classified using an ensemble of a SVM and a weighted k-NN. A classical SVM, the original restricted stacking (R-STACK) and the current improved method (EVOR-STACK) were compared. The results showed the evolutive approach obtained the best results in the context of the real data from a riparian area of Huelva (Spain).

Even though the results are satisfactory, there are still important problems to fix. Imbalanced data is the most important one. Remote sensing data provide a clear example in which the risk of dealing with imbalanced data is very high. Therefore, specific approaches for this problem will have to be taken into account to improve the final results. Finally, some problems are inherent in pixel-oriented approaches, such as the detection of partial artificial structures. In the future, it would be very interesting to apply a prior phase in which, at low addition to the computational cost, an object-oriented segmentation and classification could be carried out. In this way, the most difficult structures could be extracted and classified by means of recognition techniques from the computer vision world.

References

[1] J. Anderson, L. Plourde, M. Martin, B. Braswell, M. Smith, R. Dubayah, M. Hofton, B. Blair, Remote Sensing of Environment 112 (2008) 1856–1870.
[2] M. Andrew, S. Ustin, Remote Sensing of Environment 112 (2008) 4301–4317.
[3] D. Mason, M. Horritt, J. Dall'Amico, T. Scott, P. Bates, IEEE Transactions on Geoscience and Remote Sensing 45 (2007) 3932–3943.
[4] T. Erdody, L. Moskal, Remote Sensing of Environment 114 (2010) 725–737.
[5] P.M. Atkinson, International Journal of Applied Earth Observation and Geoinformation 5 (2004) 277–291.
[6] E.W. Bork, J.G. Su, Remote Sensing of Environment 111 (2007) 11–24.
[7] D. Mazzoni, M.J. Garay, R. Davies, D. Nelson, Remote Sensing of Environment 107 (2007) 149–158.
[8] G. Shao, J. Wu, Landscape Ecology (2008) 505–511.
[9] E.O. Tomppo, C. Gagliano, F.D. Natale, M. Katila, R.E. McRoberts, Remote Sensing of Environment (2009) 500–517.
[10] H. Fang, S. Liang, A. Kuusk, Remote Sensing of Environment 85 (2003) 257–270.
[11] A. Abraham, E. Corchado, J.M. Corchado, Neurocomputing 72 (2009) 2729–2730.
[12] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, Inc., New York, 1995.
[13] V.N. Vapnik, Statistical Learning Theory, Wiley-Interscience, 1998.
[14] F. Holger, B. Schölkopf, O. Chapelle, International Journal on Artificial Intelligence Tools (2003) 142–148.
[15] Z. Chunhong, J. Licheng, Intelligent Control and Automation 2 (2004) 1869–1872.
[16] T. Howley, M.G. Madden, Artificial Intelligence Review 24 (2005) 379–395.
[17] C. Huang, C. Wang, Expert Systems with Applications 31 (2006) 231–240.
[18] M. Caudill, AI Expert 5 (1990) 43–47.
[19] A.J. Skinner, J.Q. Broughton, Modelling and Simulation in Materials Science and Engineering 3 (1995) 371–390.
[20] D. Fogel, Neurocomputing 42 (2002) 69–86.
[21] A.D. Brown, H.C. Card, Neural Processing Letters 10 (1999) 223–229.
[22] M. Komosinski, K. Krawiec, Artificial Intelligence in Medicine 19 (2000) 25–38.
[23] H. Ahn, K. Kim, I. Han, Expert Systems 23 (2006) 127–144.
[24] S. Ozsen, S. Gunes, Expert Systems with Applications 36 (2009) 386–392.
[25] A. Herrero, E. Corchado, M.A. Pellicer, A. Abraham, Neurocomputing 72 (2009) 2775–2784.
[26] G.J. Briem, J.A. Benediktsson, J.R. Sveinsson, IEEE Transactions on Geoscience and Remote Sensing 40 (2002) 2291–2299.
[27] J. Ham, Y. Chen, M.M. Crawford, J. Ghosh, IEEE Geoscience and Remote Sensing Letters 43 (2005) 492–501. doi:10.1109/TGRS.2004.842481.
[28] J. Garcia-Gutierrez, D. Mateos-Garcia, J. C. Riquelme, in: HAIS 2010, pp. 493–500.
[29] D. Mateos-García, J. García-Gutiérrez, J.C. Riquelme-Santos, in: HAIS 2010, pp. 272-279.
[30] F.J. Cortijo, N.P.D.L. Blanca, International Journal of Remote Sensing 19 (1998).
[31] N.C. Oza, K. Tumer, Information Fusion 9 (2008) 4–20.
[32] B. Hofle, N. Pfeifer, ISPRS Journal of Photogrammetry and Remote Sensing 62 (2007) 415–433.
[33] L. Gonçalves-Seco, D. Miranda, R. Crecente, J. Farto, in: Spatial Accuracy, 2006, pp. 169–180.
[34] A.T. Hudak, N.L. Crookston, J.S. Evans, D.E. Halls, M.J. Falkowski, Remote Sensing of Environment 112 (2008) 2232–2245.
[35] A. Antonarakis, K. Richards, J. Brasington, Remote Sensing of Environment (2008) 2988–2998.
[36] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, SIGKDD Explorations 11 (2009).
[37] L.J. Eshelman, J.D. Schaffer, Foundation of Genetic Algorithms (1993) 187–202.
[38] S. Keerthi, S. Shevade, C. Bhattacharyya, K. Murthy, Neural Computation 13 (2001) 637–649.
[39] S. Garcia, D. Molina, M. Lozano, F. Herrera, Journal of Heuristics 15 (2009) 617–644.
[40] S. Garcia, A. Fernandez, J. Luengo, F. Herrera, Information Sciences 180 (2010) 2044–2064.
[41] J. Luengo, S. García, F. Herrera, Expert Systems with Applications 36 (2009) 7798–7808.
[42] S. Garcia, F. Herrera, Journal of Machine Learning Research 9 (2008) 2677–2694.
[43] L. Bruzzone, S. Serpico, Pattern Recognition Letters 18 (1997) 1323–1328.
[44] D. Williams, V. Myers, M. Silvious, IEEE Geoscience and Remote Sensing Letters 6 (2009) 528–532.
[45] R. Alaiz-Rodriguez, A. Guerrero-Curieses, J. Cid-Sueiro, Journal of Machine Learning Research 8 (2007) 103–130.
[46] A. Fernández, M.J. del Jesus, F. Herrera, Expert Systems with Applications 36 (2009) 9805–9812.

[47] Z. Zhao, P. Zhong, Y. Zhao, Mathematical and Computer Modelling in press. doi:10.1016/j.mcm.2010.11.041.
[48] S.-C. Lin, Y.-C.I. Chang, W.-N. Yang, Neurocomputing 73 (2009) 484–494.