Iterative Hyperspectral Image Classification Using Spectral–Spatial Relational Features

Pietro Guccione, Luigi Mascolo, and Annalisa Appice

***Abstract*—This paper describes the principles and implementation of an algorithm for the classification of hyperspectral remote sensing images. The proposed approach is novel and can be included within the category of the spectral–spatial classification algorithms. The elements of novelty of the algorithm are as follows: 1) the implementation of two classifiers that work iteratively, each one exploiting the decision of the other to improve the training phase, and 2) the use of relational features based on the current labeling and on the spatial structure of the image. The two classifiers are fed with the spectral features and with the spatial features, respectively. The spatial features are built using the relative abundance of each class in a neighborhood of the pixel (homogeneity index), where the neighborhood is properly defined. An important contribution to the success of the method is the adoption of a multiclass classifier, the multinomial logistic regression, and a proper use of the posterior probabilities to infer the class labeling and build the relational data. The results of the two classifiers are eventually combined by means of an ensemble decision. The algorithm has been successfully tested on three standard hyperspectral images taken from the Airborne Visible–Infrared Imaging Spectrometer and ROSIS airborne sensors and compared with classification algorithms recently proposed in the literature.**

***Index Terms*—Hyperspectral image classification, iterative classification, Markov random field (MRF), multinomial logistic regression (MLR), spectral–spatial analysis.**

I. Introduction

REMOTE sensing hyperspectral imaging sensors are spaceborne or airborne instruments able to collect hundreds of images at the same time and for the same area of the Earth, each one corresponding to a different wavelength channel [1]. The concept was first introduced by the NASA Jet Propulsion Laboratory with a system called Airborne Imaging Spectrometer and successively continued using the Airborne Visible–Infrared Imaging Spectrometer (AVIRIS), which is able to collect more than 200 spectral bands in the visible and near infrared wavelengths [2].

A hyperspectral image is a *cube* of images where each pixel is a vector of values representing the spectral signal that characterizes the underlying object. The number of applications of such systems is large and increasing [3]. One of the prominent applications is classification, i.e., the ability to generate a thematic map where each pixel is assigned to a specific class of land cover [4], [5]. The classification of hyperspectral images is a challenging problem for several reasons: 1) The spectral signature given by an area (like bare soil, vegetation, and cropping) is just a theoretical reference since the footprint of a resolution cell is actually a function of many factors due to illumination (acquisition geometry) and weather conditions (period of the year and meteorological status); 2) each resolution cell imaged by the sensor usually includes more than a single land cover (called *endmember* in [6]); and 3) the spectral characteristics of an endmember are subjected to time and spatial changes.

The problem of hyperspectral image classification has been faced using different approaches. In principle, good classification performance can be achieved using standard classification algorithms if a large number of training pixels (properly distributed among the classes) is available and the test set is generated using pixels taken from the same image. However, it is difficult to define a proper training set for the learning of the classification algorithm [7]. The effort needed to collect such pixels that often requires a human-supervised work, as well as the need to extend the algorithm to other images with the same (or similar) class distribution, makes the problem still open and challenging.

Often, in practical applications, the number of labeled pixels is not sufficient to perform a reliable estimate of the classifier parameters in the learning phase of the algorithm. If the number of training samples is relatively small compared to the number of features (and to the parameters to be estimated), we have the well-known Hughes phenomenon [8]. Another critical aspect is the quality of the training samples, affected by the nonstationary nature of the spectral signatures of the classes in the spatial domain and by the correlation existing among them [9]. The nonstationarity of the features in the spatial domain is due to physical factors such as the characteristics of the imaged area and the atmospheric conditions at the time of acquisition. In principle, a training set should be able to catch the whole statistics behavior of the scene, but this is unfeasible, as it only implies the prior knowledge of the statistical properties of the scene. On the other hand, the correlation between pixels violates the assumption of independence between samples, thus reducing the amount of information conveyed to the classification algorithm.

To overcome the problem of limited quantity and quality of training samples, some basic approaches have been proposed in the literature. Among the best performing supervised learning

Manuscript received July 14, 2014; revised October 1, 2014 and November 3, 2014; accepted December 3, 2014.

P. Guccione and L. Mascolo are with the Dipartimento di Ingegneria Elettrica e Informazione, Politecnico di Bari, 470125 Bari, Italy (e-mail: pietro.guccione@poliba.it; luigi.mascolo@poliba.it).

A. Appice is with the Dipartimento di Informatica, Universitá degli Studi di Bari, 470125 Bari, Italy (e-mail: appice@di.uniba.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TGRS.2014.2380475

methods, support vector machines (SVMs) have been largely applied with success in hyperspectral image classification [10], [11]. SVMs are also known as maximum margin classifiers, as they define a separating hyperplane that maximizes the distance between the nearest data points on both sides. In SVMs, only a subset of the training samples (the support vectors) contributes to the classification rule. Furthermore, in order to face the problem of nonlinearly separable classes, the kernel trick can be exploited. It allows us to solve the problem of defining a nonlinear separating hyperplane in a higher-dimensional feature space by performing a nonlinear mapping of the data, where the separating hyperplane is linear. As a consequence, the kernels allow us to train a linear machine in a nonlinear space, potentially circumventing the high-dimensional feature problem inherent in the hyperspectral image classification. In several works, SVMs have proved to perform better compared to other classification algorithms, including artificial neural networks [12]. SVMs are eventually less sensible to the Hughes phenomenon; on the other hand, they are intrinsically binary classifiers, and the tricks adopted to circumvent this impairment, such as the one versus all, one versus one, or hierarchical approach, do not definitively solve the problem as a multiclass classifier does.

Other attractive approaches to classification are those based on transductive learning (TL) [9], [13], [14] and active learning (AL) [7], [15]. In the TL, an iterative algorithm is implemented, and both labeled and unlabeled samples are used in the classification process. At each iteration, a number of unlabeled samples are optimally chosen to condition the hyperplane position and shape and to provide a better classification of the remaining samples in the next iteration. TL is especially effective in cases where the collection of a reliable training set (in terms of spatial and statistical distribution and of size) can be very demanding. In AL, classification is achieved in an iterative process, where, at each step, few unlabeled samples are added to the training set (as for the transductive). These samples are chosen in an optimal way, but supervision is needed to assign them to a class [7], [15]. The process is iterated, producing a clear improvement of the classification accuracy but requiring, at the same time, a large effort from an external supervisor.

Another set of strategies followed in hyperspectral image classification concerns the exploitation of the spatial information, as near pixels are supposed to be correlated to one another. The success of such techniques arises from the fact that images cannot be treated as an unordered listing of spectral measurements without spatial arrangement [16]–[18], so labeled samples can support the classification of their neighbors. One of the approaches used to integrate spatial and spectral information consists in modeling the image with a Markov random field (MRF). The spatial-contextual information is included in the classification process by adopting a maximum likelihood prior to modeling the image of class labels [18], [19]. This prior model encourages piecewise segmentations, fostering solutions in which close pixels are likely to belong to the same class. This approach can be considered a generalization of the Ising model (with Gibbs distribution) [20] successfully adopted in many image segmentation problems [18] to find the optimal solution as the one with the minimum inner energy.

In this paper, we use the central strategy of both the approaches, i.e., the iterative process and the spatial information. The central idea that we take from the iterative approach (as in TL and AL) is that it is possible to start with a small set of training samples. This set is enlarged using information coming from an increasing number of unlabeled samples considered reliable enough to contribute to the estimation of the classifier parameters for a new learning session. In principle, the reliability of the classification is expected to increase at each iteration. On the other hand, the contextual information that can be extracted from a label field (derived from a previous classification process) allows us to increase the accuracy since already classified samples can be used to infer a new decision on unreliable pixels. This idea is not too far from the loopy belief propagation approach [21]–[23], where marginals of class distribution are computed by iteratively passing a message between samples, becoming more and more reliable at each passage.

The two strategies are combined using two different classifiers: one learned from the spectral features and the other from the relational features. The two classifiers improve each other using the output of the previous iteration. A native multiclass classifier, the multinomial logistic regression (MLR), is used for both classifiers, and both the soft (i.e., the posterior probability of each class) and hard (i.e., the labeling) decisions for each unlabeled sample are exploited in the process. The posterior probabilities of the two classifiers are eventually combined using an ensemble decision [24] to achieve the joint class prediction and decide the exit strategy from the iterative loop. The relational features are built using labeled samples to get the homogeneity index [25], i.e., the relative abundance of a class in a neighborhood of a pixel. The definition of the neighbor may be function of spatial and spectral characteristics of the image, and possibly may include nonlinear transformations of the image through morphological operators [26].

This paper is organized as follows. In Section II, the problem is approached, and the key concepts of the algorithm are outlined. In Section III, the elements of the problem are given, and in Section IV, the algorithm is described. Section V reports the experimental results and the relative discussion. Finally, Section VI draws the conclusions and discusses future developments.

II. Problem Approach

A. CC Approach

Collective classification (CC) refers to the combined classification of a set of interlinked objects using information coming from the relation existing between features, between labels and features, or between labels and labels of neighbor objects [27]. CC increases the classification accuracy when the class labels of inter-related objects are correlated [24], [28], [29]. An effective way to exploit interlinked relations among pixels in a hyperspectral image is to use the information coming from the given features (the pixel spectral signature, called *intrinsic*) and from a new set of features, the *relational features*, that summarize the labeling achieved from the neighborhood. Usually, spatial neighborhood is the most common way to exploit the relation among pixels in an image, as pixels that are spatially close

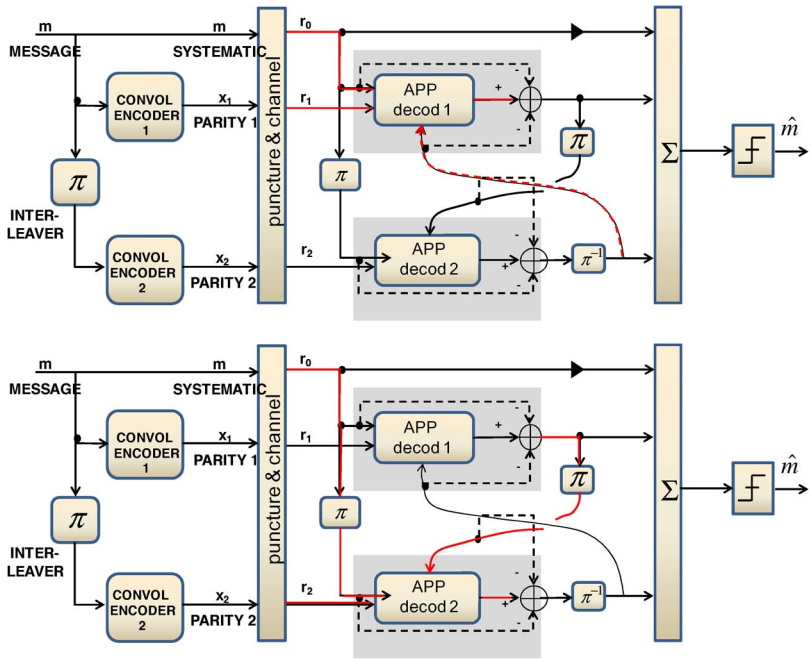

Fig. 1. High-level scheme of turbocode encoder and decoder. (a) First decoder (APP decod 1) exploits the intrinsic information and the first parity passed through the channel. (b) Second decoder (APP decod 2) uses the intrinsic information, the second parity (passed through the channel), and the output of the first APP decoder. From the second iteration on, the first decoder exploits also the extrinsic information given as output of the second decoder. The process then iterates. For a complete explanation of turbocodes working, refer, for example, to [35].

between them are most likely to spectrally behave in the same way, and this should be reflected in the relational features. On the other hand, the construction of relational features based on labeled samples is another way to build an MRF model of the image.

In the classical works on CC, labeling is performed using the joint information retrieved from intrinsic and relational features [29]. CC methods explicitly use the data relationship to build additional features exploited to learn a more predictive classifier. One of the known problems in CC is that, usually, most of the labels are unknown. A way to overcome this problem is to refine classification through iteration using algorithms as belief propagation, Gibbs sampling, or iterative classification algorithm (ICA) [22], [27].

In ICA, as an example, the iterations are initialized by labeling the samples of the test set. To this aim, a classifier learned on the training set only (*bootstrap classifier* [24]) is used. Next, the relational features are built using both the known and the predicted labels; then, the classifier re-predicts labels for the test set using both intrinsic and relational features. The process is iterated until convergence or a stop criterion is reached. The ICA algorithm is known to suffer from some issues. First, relational features are computed by aggregating classes collected in a neighborhood with a specific fixed size, while real networks are frequently characterized by a class distribution which may vary in shape and density across the network. Second, the number of examples in the training set is never extended through the iterative process. Finally, the class labels are predicted by a single classifier, while the ensemble of classifiers is frequently more accurate [30].

For this reason, a number of algorithms belonging to the class of semisupervised learning algorithms (SSLs) have recently appeared in the literature [31]–[34]. They perform a further external iterative loop in which the learning is repeated every-time the relational features are updated at the end of an ICA process. These algorithms are distinguished for the way they use the relational features to update the classifier (taking the ones computed on all the samples or just on the training set).

B. From Collective Classification to a "Turbo" Classifier

The central strategy proposed in this paper starts from the SSL algorithms presented in [24] but introduces a fundamental difference: Here, we use two classifiers, one learned from the intrinsic features and the other from the relational features. The second classifier can be learned only once the first classifier has labeled all the pixels since the relational features require the labeling of all the pixels to be defined. Once the two classifiers have been learned the first time, their output can be used to increase the training set of the other (cross-iterative process), exploiting the most reliable predicted samples. At each iteration, the two classifiers calculate the posterior probabilities of each class. These posteriors are used to both extend the training set and to learn the other classifier at the successive iteration. The reliability of the samples is given by the posterior probability distribution among the classes: the more uniform such distribution is, the less reliable a sample is.

Even if not strictly necessary to the understanding of the concept, we consider useful to show the analogy between the algorithm approach and the iterative decoding used in the turbocodes [35]. The turbocodes are a class of high-performance forward error correction codes, where the encoded binary message is composed by three concatenated parts [see Fig. 1(a)]: The information bits sent in clear (m) and two different parities

(x_1 and x_2), achieved as output of two convolutional encoders. Decoding is performed using an *a posteriori* probability (APP) algorithm, the Bahl, Cocke, Jelinek, and Raviv algoritm [36], applied on the intrinsic information (r_0, i.e., the clear part of the message passed through the channel) and on the extrinsic information, which is r_1 (i.e., the first parity passed through the channel), and the output of the second APP decoder, when available [from the successive iterations; see red arrows in Fig. 1(a)]. The output of the first APP decoder, together with the parity of the second encoder (passed through the channel r_2), represents the extrinsic information of the second APP decoder [see Fig. 1(b)]. The output of each decoder is iteratively used to improve the decoding of the other until a convergence criterion is reached and the estimation $\hat{m}$ of the message is given. The turbocodes are powerful error correcting codes, reaching stability in few iterations and being able to correct large bursts of errors. The success of turbocodes in decoding a binary message is based on the *thin distance spectrum* principle [37]. Shortly, in turbocode encoding, it is very unlikely to have binary words with low Hamming weight from both the convolutional encoders (a high weight is necessary for the success in error detection and correction); in the same way, if we supposed a good degree of independence between the spatial distribution of land cover and land cover spectral properties, the use of both properties would unlikely make both classifiers produce so different (and wrong) responses. An initial soft classification, i.e., a posterior probability, even slightly favorable to the correct class will be improved by iterations. Classification errors may occur only when a wrong class is assigned initially to both the classifiers; in that case, the iterations can amplify the effect.

The analogy of the proposed algorithm with turbocode decoding structure can be appreciated comparing Figs. 1 and 4.

III. Algorithm Description

A. Problem Definition

Let us suppose having a hyperspectral image composed of N pixels, $\mathcal{V} = \{v_1, \ldots, v_N\}$ (whose location in the image is known), characterized by a M-dimensional vector of features $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, with $\mathbf{x}_n^T = [x_{n1}, \ldots, x_{nM}]$. The features are the measurements (taken from a resolution cell) in the spectral bands acquired by the sensor, possibly after the removal of the most noisy bands (for example, the bands of water vapor absorption). We further suppose that the set of pixels can be partitioned into K unordered classes, or subpopulations, denoted by $\mathcal{C} = \{\Pi_1, \ldots, \Pi_K\}$. Each pixel in $\mathcal{V}$ is supposed to be classified into one and only one of such classes.

The purpose of the hyperspectral image classification problem is to infer the class label of a set of unclassified pixels when it is known: 1) the value of the features for all the pixels, i.e., the random vector $\mathbf{X}$, $\forall v_n \in \mathcal{V}$, and 2) a limited set of known labels, i.e., for a small number of pixels (the so-called training set), $\mathcal{V}^{(a)} = \{v_1, \ldots, v_L\} \subset \mathcal{V}$ [(a) stands for assigned], the classes are known. Typically, $L \ll N$.

Usually, the remaining samples, i.e., $\mathcal{V}^{(u)} = \mathcal{V} \setminus \mathcal{V}^{(a)}$, are considered the test set, for which the label must be assigned.

B. MLR

In the MLR model, the posterior probability of the classes satisfies the following relation [38]:

$$p(v_n \in \Pi_i | \mathbf{X} = \mathbf{x}) = \frac{f_i(\mathbf{x})\pi_i}{\sum_{k=1}^{K} f_k(\mathbf{x})\pi_k} \tag{1}$$

with π_k being the prior probability for the class Π_k, $\pi_k = p(\Pi_k)$, and

$$f_k(\mathbf{x}) = p(\mathbf{X} = \mathbf{x} | v_n \in \Pi_k) \tag{2}$$

being the likelihood. Equation (1) can be rewritten by setting the following log-odd functions [39]:

$$u_i(\mathbf{x}) = \log\left(f_i(\mathbf{x})\pi_i\right) \tag{3}$$

with which the posterior probabilities can be written as follows:

$$p(v_n \in \Pi_i | \mathbf{x}) = \frac{\mathrm{e}^{u_i(\mathbf{x})}}{\sum_{k=1}^{K} \mathrm{e}^{u_k(\mathbf{x})}}, \quad i = 1, \ldots, K. \tag{4}$$

The idea behind the MLR model is to construct a predictor that sets up a score function starting from a set of weights linearly combined with the explanatory variables (i.e., the features) of a given observation. The usual assumption for the conditional probabilities of the features is that they are distributed as a multivariate Gaussian, i.e.,

$$f_k(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{5}$$

so that the log-odd ratios

$$L_i(\mathbf{x}) = u_i(\mathbf{x}) - u_K(\mathbf{x}) = w_{0,i} + \mathbf{w}_i^T \cdot \mathbf{x} \tag{6}$$

are linear combinations of the features. In the MLR model, the *log-odd* ratios are linear combinations of the features, but the Gaussian assumption is removed. This way, the weights are no longer explained through mean and covariance; rather, they are estimated straightly through $\mathbf{X}$.

In this paper, the posterior probability that a sample belongs to a given class is modeled using the MLR, i.e.,

$$p(v_n \in \Pi_i | \mathbf{x}) = \frac{\mathrm{e}^{L_i(\mathbf{x})}}{1 + \sum_{k=1}^{K-1} \mathrm{e}^{L_k(\mathbf{x})}}, \quad i = 1, \ldots, K-1$$

$$p(v_n \in \Pi_K | \mathbf{x}) = \frac{1}{1 + \sum_{k=1}^{K-1} \mathrm{e}^{L_k(\mathbf{x})}} \tag{7}$$

with $L_i(\mathbf{x})$ defined in (6). The model can be directly extended to the kernel version of MLR, in which the weights are combined with a set of functions of the input features [40], [41].

Usually, in an MLR classifier, the class assigned to a sample is the one for which the probability is the largest among the classes, i.e.,

$$\hat{y}_{nk} = \begin{cases} 1 & k = \arg\max\limits_{k} \{p(v_n \in \Pi_k | \mathbf{x}_n)\} \\ 0 & \text{elsewhere.} \end{cases} \tag{8}$$

To fit the model (i.e., to learn the classifier), the set of $(M + 1) \cdot (K - 1)$ weights needs to be estimated (see Appendix A).

1	2	2	2	2	3	3	3	3	1	1	3	1	2	2
2	2	2	2	2	2	3	2	3	3	3	3	2	2	2
2	2	2	2	2	3	3	2	2	1	1	1	1	1	1
3	3	3	3	3	3	2	2	2	1	1	3	1	1	1
3	3	1	1	3	3	3	2	2	3	3	3	1	1	2
3	1	1	1	1	1	3	3	2	3	1	1	3	3	2
1	1	1	1	1	3	3	2	2	3	1	1	3	3	2
3	2	2	2	2	1	1	3	3	3	2	2	3	3	2
2	1	1	1	1	1	1	1	2	2	2	2	2	2	2
1	1	3	3	3	2	2	2	2	3	3	2	2	2	2
1	1	3	3	3	3	2	2	1	1	1	2	3	3	1
2	3	3	1	2	2	2	2	1	1	1	2	3	3	1
3	3	3	2	2	2	2	2	1	1	1	2	3	3	1
2	2	3	2	2	2	2	2	2	1	3	3	3	3	1
2	2	3	2	2	2	2	2	2	2	3	2	3	3	3

Fig. 2. In the example, the SDHI is computed for the central pixel (in red) to which class 3 is currently assigned. The pixels in green belong to the window used to compute the SDHI according to (9).

C. Relational Features

In the following, we suppose that the image has been already labeled, i.e., the matrix of the target variables $\mathbf{Y} = [y_{nk}]$, $\forall n, k = 1, \ldots, K$, is someway given. There are a number of possibilities to construct the relational features from the labels using spatial or combining spatial and spectral properties of the image. However, we have considered a too dispersive listing of all the possibilities, as a sensitive analysis of the parameters related to the method would follow in the experimental results. Instead, we here describe just one method, called the spatial-dependent homogeneity index (SDHI).

For a given pixel v_n, let us define ∂v_n as the set of pixel neighbors of v_n, including v_n (the neighborhood is properly defined hereinafter). The SDHI is defined as

$$\alpha_{nk} = \frac{|\partial v_n|_{v_j \in \Pi_k}}{|\partial v_n|}, \quad k = 1, \ldots, K \tag{9}$$

i.e., the relative number of pixels that belong to a given class Π_k in the neighbor set ($|\partial v_n|$ is the cardinality of the neighborhood ∂v_n that must not be a null set). If we define a collection of neighbor sets as follows:

$$\left\{ \partial v_n^{(s)} \right\}, \quad s = 1, \ldots, S \tag{10}$$

we have $S \cdot K$ relational features defined as in (9). The shape of the neighbor set can have the shape of either a square or a circle with radius W or can have other shapes (for example, we can consider the shapes adopted as structuring elements in the morphological operations [42]); in alternative, particular shapes can be used if prior information is available.

To simplify, let us suppose having the (partial) label map illustrated in Fig. 2 as the starting condition to compute the SDHI of the central pixel (in red) where only three classes are present (enumerated by 1, 2, and 3). If we choose a square neighborhood of size $W = 7$ (in green), the SDHIs for the three classes are $\alpha_1 = 14/49$, $\alpha_2 = 16/49$, and $\alpha_3 = 19/49$. Please note that $\alpha_1 + \alpha_2 + \alpha_3 = 1$.

1: **procedure** $\mathcal{F}(\mathcal{X}, \mathcal{Y}^{(a)}, \epsilon, \beta, \eta_1, \eta_2, N_{it})$
2: $i = 0, \ \Delta G = 2\epsilon, \ \mathcal{Y}_{(1)} = \mathcal{Y}^{(a)} \ \mathbf{X}_{(1)} = \mathbf{X}^{(a)}$
3: **while** $i < N_{it} \ \ AND \ \ \Delta G > \epsilon$ **do**
4: $\mathbf{W}_x = \text{train}\left(\mathbf{X}_{(1)}, \mathcal{Y}_{(1)}\right)$
5: $p_{(1)} = \text{findP}(\mathbf{X}, \mathbf{W}_x)$
6: $\mathcal{Y}_{(1)} = \text{classify}(p_{(1)}, 0)$
7: $G_1 = \text{negllh}(p_{(1)}, \mathcal{Y}_{(1)})$
8: $\mathcal{Y}_{(1)}(\mathcal{V}^{(a)}) = \mathcal{Y}^{(a)}$
9: $\mathbf{X}_R = \text{relational}(\mathcal{Y}_{(1)}, \mathbf{X})$
10: $\mathcal{Y}_{(2)} = \text{classify}(p_{(1)}, \beta \cdot \mathrm{e}^{-\eta_1 i}))$
11: $\mathcal{Y}_{(2)}(\mathcal{V}^{(a)}) = \mathcal{Y}^{(a)}$
12: $\mathbf{W}_r = \text{train}\left(\mathbf{X}_{R(2)}, \mathcal{Y}_{(2)}\right)$
13: $p_{(2)} = \text{findP}(\mathbf{X}, \mathbf{W}_r)$
14: $\mathcal{Y}_{(2)} = \text{classify}(p_{(2)}, 0)$
15: $G_2 = \text{negllh}(p_{(2)}, \mathcal{Y}_{(2)})$
16: $\mathcal{Y}_{(1)} = \text{classify}(p_{(2)}, \beta \cdot \mathrm{e}^{-\eta_2 i})$
17: $\mathcal{Y}_{(1)}(\mathcal{V}^{(a)}) = \mathcal{Y}^{(a)}$
18: $G^{(i+1)} = \sqrt{G_1 \cdot G_2}, \ \ \Delta G = G^{(i+1)} - G^{(i)}$
19: $i = i + 1$
20: **end while**
21: $p_{en} \propto p_{(1)} \cdot p_{(2)} / p(\mathcal{Y})$
22: $\hat{\mathcal{Y}} = \text{classify}(p, 0)$
23: **end procedure**

Fig. 3. IRMC algorithm for hyperspectral image classification.

IV. Algorithm Implementation

The Iterative Relational Multinomial Classifier (IRMC) algorithm is illustrated in Fig. 4 and formally described in Fig. 3.

A. Algorithm Steps

The inputs of the algorithm are the spectral features $\mathcal{X}$, the labels of the training set $\mathcal{Y}^{(a)}$, the number of classes K, the pixel locations $\mathcal{V}$ (used to define the neighbor sets), and a few real-valued parameters detailed in the following. The algorithm steps are described in the following.

1) The algorithm is initialized by using the feature vectors in the training set $\mathbf{X}_{(1)} = \mathbf{X}^{(a)}$ and their corresponding labels $\mathcal{Y}_{(1)} = \mathcal{Y}^{(a)}$ (line 2 of the algorithm). This training set shall be gradually extended during iterations.
2) In the conditional loop (line 3), the first classifier is trained (the function `train`) using the spectral features and the associated labels (line 4). For an MLR classifier, learning means to calculate the set of weights, $\mathbf{W}_x$, as described in Appendix A.
3) The first classifier is then used to find the posterior probabilities for the test set using (7), which corresponds to the function `findP` (see line 5).
4) The posterior probabilities $p_{(1)}$ are used to label the image (i.e., the samples of the test set, as the training set is already labeled), $\mathcal{Y}_{(1)}$, as in (8) (function `classify`; see line 6). The labels for the pixels of the training set are left unchanged (line 8).
5) The posterior probabilities $p_{(1)}$, and if necessary the spectral features, are used to build the relational features $\mathbf{X}_R$ as in (9) (line 9).

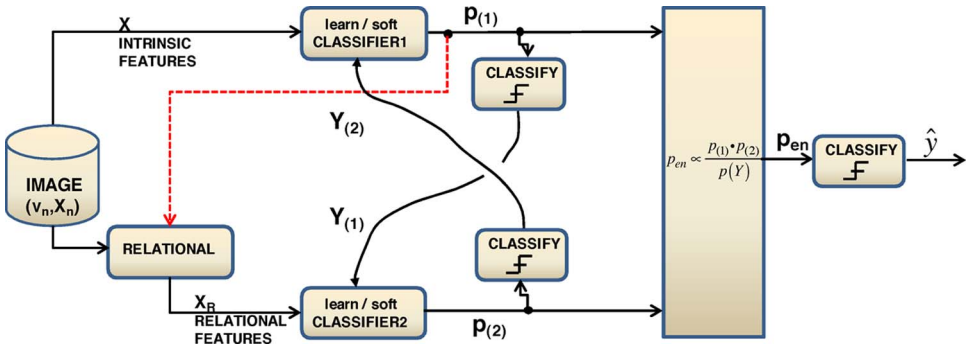

Fig. 4. Classification algorithm block diagram. Please note the analogy in the structure with the turbocode decoding scheme proposed in Fig. 1. The role of the intrinsic information is here played by the spatial coordinates of pixels, while the extrinsic information (r_2 in Fig. 1) are generated using the output of the first classifier (red dotted line), so it is not a totally independent source coming from the channel, as for turbocodes. This is the main difference between the two algorithms.

6) The second classifier is learned (line 12) using the enlarged training set and their corresponding relational features $\mathbf{X}_{R(2)}$. The pixels of this set are defined as the ones for which the maximum probability among the classes is higher than a threshold

$$\mathcal{V}_{(2)} = \left\{ v_n | y_{nk} = \left\{ \max_{k \in \mathcal{C}} \left\{ p_{(1),nk} \right\} > \beta \cdot \mathrm{e}^{-\eta_1 i} \right) \right\} \quad (11)$$

with $\beta > 0.9$ (line 10). The threshold is progressively relaxed to increase the set size during the iterations i and to account for the supposed progressive convergence of the algorithm towards a stable solution. As in point 4), the labels of the training set are left unchanged (line 11). The output of the learning stage is the set of weights $\mathbf{W}_r$.

7) The second classifier is then used to find the posterior probabilities $p_{(2)}$ for all the pixels [(7), line 13].
8) From the posterior probabilities $p_{(2)}$, the image is labeled again, $\mathcal{Y}_{(2)}$ [(8), line 14].
9) To close the loop, the new training set processed by the first classifier is built using a relation similar to (11), where, again, the threshold is progressively relaxed (line 16)

$$\mathcal{V}_{(1)} = \left\{ v_n | y_{nk} = \left\{ \max_{k \in \mathcal{C}} \left\{ p_{(2),nk} \right\} > \beta \cdot \mathrm{e}^{-\eta_2 i} \right\} \right\}. \quad (12)$$

The parameters η_1 ad η_2 are chosen experimentally (with $\eta = 0$, meaning that no relaxation is given at all). Again, the label for the pixels of the training set is left unchanged (line 17).

10) In the literature [38], the negative logarithm of the likelihood is usually defined as the error function. The error function, computed at the end of the classification process for the first and the second classifiers (lines 7 and 15, respectively, and function `negllh`), is used to evaluate the degree of convergence of the cycle

$$G = -\sum_n \sum_k y_{nk} \log p_{nk}, \quad n = 1, \ldots, N. \quad (13)$$

Since we have two error functions (G_1 and G_2), their geometric mean is taken, and its difference with respect to the previous iteration (ΔG) is used to condition the loop (lines 18, 19, and 3).

B. Final Class Labeling

At the end of the iterative process, the outcome of the algorithm is a set of labels describing the whole image and obtained as the output of the first and second classifiers $\mathcal{Y}_{(1)}$ and $\mathcal{Y}_{(2)}$. Initially, the two sets of labels are mostly different since each of them has been obtained exploiting different properties of the image: the spectral similarity in one case and the spatial correlation of labels in the other. It is for this reason that the combined use of the two classifiers is expected to give the highest *information* in the first few iterations, when the diversity between the two labels is the largest. Successively and until convergence, it is expected that the labels will provide progressively similar information, i.e., the solution tends to stabilize with the increase of iterations. This reasoning is supported by the experimental results in Section V-C.

To get the joint probability, instead of $\mathcal{Y}_{(1)}$ and $\mathcal{Y}_{(2)}$, we use the posterior probabilities achieved at the end of the conditional loop, $p_{(1)}$ and $p_{(2)}$. The two conditional probabilities are the (soft) outcome of the two classifiers, given the intrinsic and the relational features, respectively

$$\begin{aligned} p_{(1)} &= p(\mathcal{V}_{(1)}|\mathbf{X}) \\ &= \left\{ p(v_n \in \Pi_k | \mathbf{X} = x_n), v_n \in \mathcal{V}_{(1)} \right\} \\ p_{(2)} &= p(\mathcal{V}_{(2)}|\mathbf{X}_R) \\ &= \left\{ p\left(v_n \in \Pi_k | \mathbf{X}_R = x_{R,n}, v_n \in \mathcal{V}_{(2)} \right) \right\}. \end{aligned} \quad (14)$$

Assuming $\mathbf{X}$ and $\mathbf{X}_R$ as two conditionally independent events, given the class $\mathcal{Y}$, the combined prediction (see line 21 and [24]) is

$$p(\mathcal{Y}|\mathbf{X}, \mathbf{X}_R) = \frac{p(\mathbf{X}|\mathcal{Y}) p(\mathbf{X}_R|\mathcal{Y}) p(\mathcal{Y})}{p(\mathbf{X}, \mathbf{X}_R)} \quad (15)$$

and after a few passages

$$p(\mathcal{Y}|\mathbf{X}, \mathbf{X}_R) = \frac{p(\mathcal{Y}|\mathbf{X}) p(\mathcal{Y}|\mathbf{X}_R)}{p(\mathcal{Y})} \cdot \frac{p(\mathbf{X}) p(\mathbf{X}_R)}{p(\mathbf{X}, \mathbf{X}_R)}. \quad (16)$$

In the previous equation, $p(\mathcal{Y}|\mathbf{X})$ and $p(\mathcal{Y}|\mathbf{X}_R)$ are $p_{(1)}$ and $p_{(2)}$, respectively, while $p(\mathcal{Y})$, i.e., the prior probability of the class distribution, can be estimated using the training set (the supervised classification of the training set is supposed to be performed on pixels picked up in a random way), or

exploiting some prior information concerning the class abundance. The second fraction does not need to be estimated since $p(\mathcal{Y}|\mathbf{X}, \mathbf{X}_R)$ can be estimated using a normalization independent of $\mathcal{Y}$. The posterior probability is hence used for the final classification (line 22) as in (8).

V. Experimental Results

The accuracy of the IRMC has been evaluated using three real hyperspectral data sets. The remainder of the section includes the following: a description of the data sets used to achieve the performances (Section V-A), a description of the experiments carried out (Section V-B), a discussion on the selection of the parameters and algorithm analysis (Section V-C), the presentation of the results (Section V-D), the sensitivity analysis (Section V-E), and the comparison with the previous literature and a discussion about the advantages and limit of the algorithm (Section V-F).

A. Data Set

The data sets used for the evaluation of the algorithm performances are Indian Pines, University of Pavia, and Salinas. These data sets are well known and widely used benchmarks for hyperspectral image classification and can be easily found on the Web.

1) AVIRIS Indian Pines: The scene was gathered by the AVIRIS sensor [2] over the Indian Pines test site in Northwestern Indiana and consists of 145 $\times$ 145 pixels and 224 spectral reflectance bands in the wavelength range 0.4–2.5 μm. The scene is a subset of a larger one and includes two-thirds agriculture and one-third forest or other natural perennial vegetation. There are other topographic elements such as two highways, a rail line and low-density dwelling, and smaller roads. Since the scene has been taken in June, some of the crops present are in their early stages of growth. The ground truth available has been organized into 16 classes not all mutually exclusive. The number of bands is usually reduced to 200 by removing bands covering the region of water absorption.

2) ROSIS Pavia University: The scene was gathered by the ROSIS sensor during a flight campaign over Pavia, northern Italy. The number of spectral bands is 103. The scene consists of 610 $\times$ 340 pixels, but some of the samples contain no information and have been discarded before the analysis. The geometric resolution is 1.3 m. The image ground truth has been differentiated into nine classes.

3) AVIRIS Salinas: This scene was collected by the AVIRIS sensor over Salinas Valley, California, and is characterized by high spatial resolution (3.7-m pixels). The area is composed by 512 $\times$ 217 pixels. As for the Indian Pines scene, the 20 water absorption bands have been discarded. The scene includes vegetables, bare soils, and vineyard fields. The ground truth has been organized into 16 classes.

B. Experimental Setting

The experiments have been executed using four different classifiers: the proposed IRMC and three basic classifiers taken for comparison, i.e., the MLR and the SVM with both linear (lSVM) and radial basis function kernel (kSVM). For SVM, the one-against-all strategy has been adopted for the choice of the final class.

TABLE I
Experiment Settings for MLR, IRMC, lSVM, and kSVM

Setting	MLR	IRMC	SVM
Reduction of feat. by PCA	yes, $\geq 99\%$		no
# Folds per experiment	5		
N_{it}	20		
ϵ	0.01		
β		0.97	
η_1		0	
η_2		0.10	
kernel rule	none		RBF, linear
size of square neighbor W		13	

The logistic regression algorithm (in MLR and IRMC) requires to iteratively solve a nonlinear system of equations to obtain the weights (see Appendix A for the solution and Appendix B for the regularization). Since some of the features are highly dependent on each other (i.e., they do not constitute a mutually exclusive minimal set of descriptors), the inversion of the matrix in (24) can be difficult, thereby making the problem ill-conditioned. The ill-conditioning problem in MLR is known and has been already faced in the literature, with code implementations publicly available on the Web [18], [43]. However, the computational complexity of such algorithms is high and makes them not suitable in the iterative scheme of IRMC. It is for this reason that we decided to reduce the set of both spectral and relational features by using the principal component analysis (retaining 99% of variance) [44].

The data set has been initially divided into two parts, a train set $\mathcal{T}$ and a test set $\mathcal{S}$, with each class population equally distributed between the two sets. Successively, smaller subsets $\mathcal{T}^{(p)}$ of data have been randomly drawn from $\mathcal{T}$, according to the experiment needs and preserving the relative class populations. Experiments have been carried out using different percentages of training sets (up to 10% of the total number of pixels in the data set, which corresponds to 20% of pixels in $\mathcal{T}$). Then, the subsets $\mathcal{T}^{(p)}$ have been used to train a classifier, and performance has been measured on $\mathcal{S}$. Each experiment has been repeated five times, and the average and standard deviation of the accuracy were reported.

No kernel rule has been applied to the logistic regression algorithms (even though it may be applied), and this is the reason to include, for a fair comparison, the linear SVM classifier. Comparison with previous studies, however, suggested to include also the kernel SVM in the analysis. For SVM algorithms, the parameters have been optimized according to a grid-search method. The relational features have been generated according to Section III-C and using a set of neighbors of square shape and increasing size.

The main experimental settings are summarized in Table I.

C. Optimization of Parameters

IRMC requires the choice of a number of parameters (see Fig. 3): $\{N_{it}, \epsilon, \beta, \eta_1, \eta_2\}$.

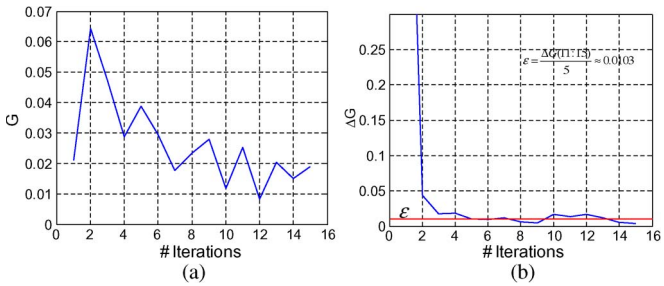

Fig. 5. Selection of the parameters that control the conditional loops. (a) Error function G [see (13)]. (b) Finite difference of the error function (ΔG in line 18) and the value chosen for ϵ (the horizontal red line). The figures refer to the AVIRIS Indian Pines data set with 10% of training set and five folds.

TABLE II
SENSITIVITY ANALYSIS OF IRMC TO PARAMETERS $\{\beta, \eta_1, \eta_2\}$ AND CORRESPONDING OA. INDIAN PINES DATA SETS, 10% OF TRAINING SET, AND FIVE FOLDS USED FOR EACH EXPERIMENT

β	η_1	η_2	OA [%]
0.90	0.1	0.1	75.93
0.93	0.1	0.1	77.08
0.97	0.1	0.1	78.04
0.99	0.1	0.1	76.43
0.97	0.1	0.02	76.21
0.97	0.02	0.1	85.73
0.97	0.01	0.1	89.15
0.97	0.0	0.1	94.06

The two parameters that control the conditional loops (the loop described in Fig. 3 and the internal loop described in Appendix A) are ϵ and N_{it}, respectively. The choice of suitable values for ϵ and N_{it} has been made basing on the trend of error function G. In an iterative process, it is expected that the error function decreases monotonically. In the case of IRMC, instead, the posterior probability out of a classifier conditions the input of the other classifier, and this explains the oscillatory trend of the error function during iterations. As already discussed in Section IV-B, by going on with iterations, it is expected that the posterior probabilities provided by the two classifiers become more similar, i.e., the corresponding labels progressively provide similar information. This reasoning is supported by the general trend of the error function of (13), which is displayed in Fig. 5(a) and which has been found similar for the other data sets.

We arbitrarily selected ϵ equal to the average, in the last five iterations, of the finite difference of the error function ΔG and N_{it} sufficiently high to be sure that ΔG has reached a steady state around the selected ϵ.

The parameters $\{\beta, \eta_1, \eta_2\}$ control the amount of transductive pixels [9] to be included at each iteration in the converging process. In detail, the exponential decaying laws in (11) and (12) control the amount of samples that are believed reliable enough to update the set of samples with which classifiers can be trained. While convergence is assured, a way to automatically derive such parameters (as in many other optimization algorithms, for example, the stochastic gradient descent) is hard to define since parameters are too much dependent on the experimental conditions.

Intuitively, a very low value of β (i.e., a very low starting threshold) yields a high number of pixels to be included too early in $\mathcal{V}_{(1)}$ and $\mathcal{V}_{(2)}$, when the differences among spectral and spatial classification are expected to be high. This decision can generate instability in the process and a lower accuracy in output. On the other hand, a too low value for η_1 or η_2 may lock the threshold to high values, avoiding the training set to enlarge and making the convergence slow.

In order to give an example, we carried out different experiments using Indian Pines with 10% of the training set, in a fivefold cross-validation. In the first set of experiments, we changed the value of β within a suitable set. Results in Table II suggest that a high value of β is desirable to achieve high accuracy, with an optimal value reached for $\beta = 0.97$. Higher values, perhaps, could make the convergence of the system too slow and drive the performance to decrease. Similarly, the values of η_1 and η_2 have been changed (letting $\beta = 0.97$), and the accuracy was measured. Final results in Table II suggest that the optimality is reached with different values among η_1 and η_2, which is reasonable, since each parameter controls the degree of convergence (during the iterations) of the labels coming from two different classifiers (spectral and spatial). The final combination of parameters used for the three data sets and for the subsequent experiments has been $\{\beta, \eta_1, \eta_2\} = \{0.97, 0.0, 0.1\}$.

D. Experimental Results

In Table III, the classifiers are compared on the three data sets in terms of overall accuracy (OA), average accuracy (AA), and κ-statistics. The results have been generated for different sizes of the training set. Linear SVM performs better than MLR in Indian Pines but not for Pavia University and Salinas, while kernel SVM performs better than linear SVM and MLR in Indian Pines and Pavia University but not in Salinas. Finally, IRMC is superior to all the classifiers and for all the percentages of the training set, even if the gain is superior for Indian Pines, more contained for Pavia University, and close to zero for the Salinas data set.

The classification maps obtained for the Indian Pines data set by IRMC and kSVM are compared in Fig. 6. A visual inspection reveals that the notable difference between the two maps is that, in SVM, the errors are uniformly distributed (the salt-and-pepper effect in the homogeneous regions) while, in IRMC, the errors are more concentrated in highly heterogeneous areas, where spatial separability among classes is more difficult. As an example, in the upper right part of the image [see Fig. 6(a)], the algorithm has confused the class Soybean-mintill with classes Corn-notill and Soybean-notill; in the upper left part of the picture, the class Corn-notill has been confused with Soybean-notill. This result is in agreement with the class accuracy in Table VII, where, for Indian Pines, we found that Corn-notill, Soybean-notill, and Soybean-mintill have a precision and a recall that are lower than the average over the classes. This is expected for Soybean-notill and Soybean-mintill as they represent different degrees of plowing or harvest of the same crops. Instead, the algorithm does not perform well for Corn-notill probably because one of the plots of land belonging to this

TABLE III
OA, AA, AND κ-STATISTICS FOR INDIAN PINES, PAVIA UNIVERSITY, AND SALINAS DATA SETS AND DIFFERENT PERCENTAGES OF TRAINING SET. LINEAR SVM, RBF KERNEL SVM, MLR, AND IRMC HAVE BEEN COMPARED

	% train	Indian Pines			Pavia Un			Salinas		
		OA [%]	AA [%]	κ	OA [%]	AA [%]	κ	OA [%]	AA [%]	κ
lSVM	3	61.76	61.78	0.5619	71.78	71.48	0.6219	89.19	89.21	0.8795
	5	66.20	66.22	0.6134	73.03	72.48	0.6378	89.55	89.56	0.8835
	10	71.18	71.19	0.6708	74.51	74.01	0.6579	90.04	90.02	0.8889
kSVM	3	69.43	69.25	0.6505	83.91	84.84	0.7839	90.00	91.81	0.8880
	5	75.29	75.31	0.7165	84.97	85.63	0.7988	90.50	91.75	0.8946
	10	80.64	80.34	0.7787	86.85	87.49	0.8248	91.69	91.70	0.9075
MLR	3	55.69	58.25	0.4936	77.60	76.41	0.6958	96.31	96.42	0.9549
	5	59.97	62.52	0.5408	78.29	77.09	0.7044	96.30	96.29	0.9548
	10	68.09	67.89	0.6335	80.87	80.07	0.7397	95.58	95.99	0.9459
IRMC	3	85.83	87.25	0.8391	87.48	87.40	0.8332	96.30	96.30	0.9548
	5	88.90	91.19	0.8743	87.98	87.84	0.8392	96.66	96.62	0.9591
	10	94.06	94.32	0.9323	88.26	88.09	0.8429	96.97	96.93	0.9630

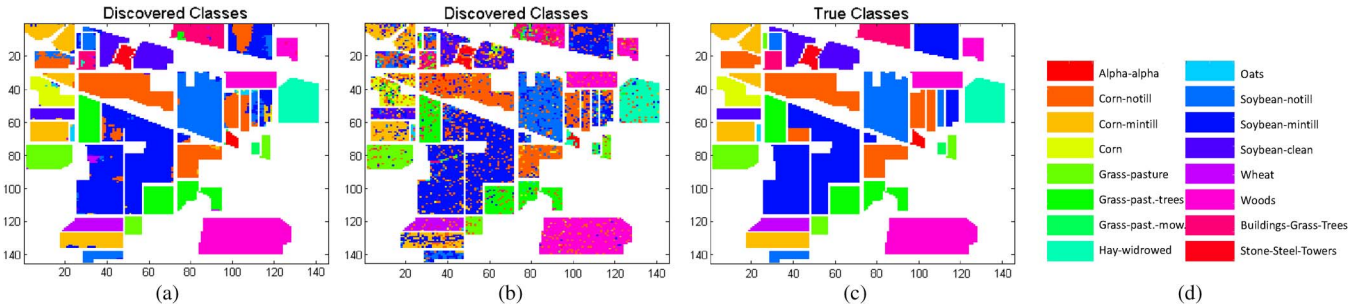

Fig. 6. Indian Pines data set classification maps for (a) IRMC (OA = 93.27%) and (b) SVM (OA = 81.84%). The ground truth and the legend are shown in (c) and (d), respectively.

class is surrounded by large areas of paths belonging to the Soybean-notill class. Even in this case, the misclassification may be due to the criterion used for the evaluation of the spatial features (shape/size of the windows).

E. Sensitivity Analysis

The key elements of IRMC are as follows: 1) the iterative procedure; 2) the use of the MLR; and 3) the supplemental information provided by the spatial features. All the three elements actively contribute to the good performances of the algorithm. To show this, we carried out further experiments in which the MLR classifier and the spatial features are removed. The experiments have been executed using the Indian Pines data set, with a training set of 10% and five folds.

In the first experiment, we replaced the classifier within the algorithm, using the naive Bayes classifier [39], since it provides also the posterior probabilities. The achieved OA was 73.29%, i.e., higher than that in the pure MLR case, but not as high as that in the IRMC case. In our version of the Bayes classifier, we adopted the multivariate normal distribution for both the spectral and spatial features, conditioned to the class. In [39], it is suggested that the logistic regression is more robust to data nonnormality than the Bayes classifier, and consequently, the performances are expected to be better. On the other hand, the logistic regression is asymptotically less efficient than Bayes and requires a higher sample set to be trained to get the same (asymptotic) performance.

TABLE IV
OA FOR DIFFERENT VERSIONS OF THE ALGORITHM

Algorithm verson	OA
pure naive Bayes	58.70
pure LR	68.09
MLR+iterations	69.79
naive Bayes+spatial+iterations	73.29
MLR+spatial+iterations(i.e. IRMC)	94.06

In the second experiment, we removed the use of the spatial information, leaving a single MLR classifier that iteratively increases its training set at each iteration. While this algorithm resembles the transductive classification, a comparison with the transductive SVM proposed in [9] cannot be given since the tSVM was tested on a data set different from those used in this study. The achieved OA has been 69.79%, which is not very different from that of the pure MLR. Results of the different tests, collected in Table IV, confirm that all the three elements are peculiar to the success of the IRMC algorithm.

We finally report the classification performance as a function of the following: 1) the size of the training set and 2) the size of the window used for the computation of the spatial features. Both such analyses have been carried on the Indian Pines data set.

In the first experiment, we tested the sensitivity of the algorithm to the size of the training set. The experimental conditions are the ones described in Section V-B. In Table V, the OA, its

TABLE V
OA, STANDARD DEVIATION OF OA, AA, AND κ-STATISTICS FOR INDIAN PINES DATA SET AS A FUNCTION OF THE TRAINING SET SIZE

Training set	OA	OA std	AA	κ
1%	78.85	10.28	82.92	0.7597
2%	80.78	6.33	82.06	0.7821
3%	85.83	4.14	87.25	0.8391
5%	88.90	3.22	91.19	0.8743
10%	94.06	2.58	94.32	0.9323

TABLE VI
OA, STANDARD DEVIATION OF OA, AA, AND κ-STATISTICS FOR INDIAN PINES DATA SET AS A FUNCTION OF THE WINDOW SIZE

S	W	OA	OA std	AA	κ
5	7,9,11,13,15	85.39	2.84	86.13	0.8336
7	7,$\cdots$,19	89.04	2.58	89.44	0.8750
9	7,$\cdots$,23	89.95	2.07	90.25	0.8853
11	7,$\cdots$,27	89.85	3.68	91.91	0.8848
13	7,$\cdots$,31	94.06	2.58	94.32	0.9323
15	7,$\cdots$,35	93.86	1.38	93.92	0.9300

standard deviation over five folds, the mean accuracy, and the κ-statistic are reported. As expected, the accuracy increases with the number of samples while the standard deviation decreases.

In the second experiment, we tested the sensitivity of the algorithm to the spatial feature extraction. The spatial features, estimated according to (9), are a function of the number of considered neighborhood windows [the value S in (10)] and of their shape and size. In order to simplify the analysis, we used a set of square windows, setting a single parameter to decide the size and number of the square windows. With reference to Table VI, for each experiment, we computed the SDHI for a set S of square windows of size $W = 2l + 7$, with $l = 0, \ldots, S-1$. Therefore, for example, with $S = 6$, we computed the SDHI for windows of size $W = [7, 9, 11, 13, 15, 17]$. The results in Table VI show that the increase of accuracy is good up to a certain value of S; it may be supposed that the performance increases until the size of neighbor windows is less or comparable with the average size of the homogeneous areas in the data set (this quantity may be someway related with the spatial correlation length). In Table V, we used the best result of Table VI, i.e., $W = 13$.

F. Comparison With Previous Studies

The result for Indian Pines is comparable with that achieved in Table II (last column) of [22], where an OA of 94.76% is reported with a training set of 10% of the ground-truth data. In Table VII, the class-by-class precision and recall (i.e., accuracy on class) have been reported. The accuracy metrics have been collected by running our algorithm with the same percentage of the training set used in [22], for a fairer comparison. The same analysis has been repeated for Pavia University, where, however, we selected a balanced training set (i.e., proportional to the class population) of 10%, while the training samples per class in Table III of [22] are variable. Anyway, for the Pavia University data set, we achieved an OA higher than that in [22] (85.78%) even with a training set of 3%. Looking through the class-by-class results of the Indian Pines data set, IRMC performs better than the algorithm proposed in [22] just for a few classes, such as *Corn-mintill*, *Grass-pasture*, or *Woods*, where we got 96.07%, 97.98%, and 97.56% of recall, respectively, compared to 90.68%, 96.66%, and 94.48% in Table II of [22]. These classes are sometimes located in areas far enough from areas belonging to other classes with which they can be confused (see in Fig. 6(c) the *Woods* class for example). A similar behavior has been found for the Pavia University data set, for example, comparing the *Bare Soil* and *Metal Sheet* classes, where we got 83.91% and 99.25% of recall, respectively, compared to 73.27% and 97.97% achieved in Table III of [22] (the ground-truth map of the Pavia data set is in [22]).

Similar considerations hold if we compare IRMC performances with the ones in Table II (last column) of [45] (for Indian Pines, 5% of the training set) and Table III (last column) of [45] (for Pavia University, the training set is chosen in a variable way, as in [22]). The IRMC class-by-class results for the case of 5% of Indian Pines are not reported; however, we found that IRMC performs worse just for the classes identified in the comparison performed above with [22]. On the other hand, the method in [45] performs better for all the classes in the Pavia University data set (in [45], the OA for the Pavia data set is 98.09%) since that algorithm has more flexibility in constructing the spatial information through an extensive use of the morphological profiles. Finally, similar considerations hold also in the comparison with the results in [30], where OA for Pavia University is higher compared with our case, while for Indian Pines, the experiment is different (in [30], a fixed number of samples per class is taken), so no conclusions can be drawn.

A general consideration on the limit of IRMC is that the proposed algorithm certainly takes advantage of the spatial information, but it has little sensitivity at the borders of homogeneous areas. IRMC has hence difficulty in the classification of pixels belonging to classes located in *crowded* (i.e., highly heterogeneous) areas, performing worse than other spatial-spectral methods. This may be due to the size of the window used to get the homogeneity index. The need to have a fixed value of such window that is large enough to capture the spatial information from neighbors is then a limit of the proposed algorithm. On the other hand, IRMC has high capability in classifying pixels in homogeneous areas once an optimized set of the spatial windows is provided (see Section V-E).

Another weakness of the algorithm, as already marked in the Introduction, is that classification errors especially occur when both classifiers assign a wrong class during the first iterations since, in that case, subsequent iterations tend to confirm the decision. A more robust version of the algorithm, in future, could take advantage of an ensemble of classifiers (for the spectral and the spatial decisions) to limit the number of these cases.

A final impairment of the algorithm is the required computational time, which substantially increases compared to simpler solutions. In order to have a comparison, a single fold of SVM on Indian Pines (experimental conditions as in Section V-D, with a training set of 5%) takes about 3.5 s (apart from the grid search, which is very time consuming), while IRMC takes about 485 s.

TABLE VII
NUMBER OF TRAINING SAMPLES, IRMC CLASS-BY-CLASS PRECISION, AND RECALL (TRAINING SET: 10% OF TOTAL DATA SET)

Indian Pines				Pavia University			
Class name	# samples	precision [%]	recall [%]	Class name	# samples	precision [%]	recall [%]
Alpha-Alpha	5	88.70	92.97	Asphalt	663	91.28	88.82
Corn-notill	143	88.24	91.19	Meadows	1865	94.81	90.78
Corn-mintill	83	96.29	96.07	Gravel	210	72.11	81.38
Corn	24	95.76	93.71	Trees	307	83.21	88.34
Grass-pasture	48	93.39	97.98	Painted metal sheets	135	98.72	99.25
Grass-trees	73	99.89	97.91	Bare Soil	503	73.13	83.91
Grass-pasture-mowed	3	94.29	76.62	Bitumen	133	61.02	74.53
Hay-widrowed	48	100.00	98.94	Self-Blocking Bricks	398	86.72	80.74
Oats	2	96.00	85.53	Shadows	95	99.75	97.74
Soybean-notill	97	91.69	89.17				
Soybean-mintill	245	92.95	94.18				
Soybean-clean	59	95.41	94.27				
Wheat	21	99.22	98.26				
Woods	126	99.34	97.56				
Buildins-Grass-Trees-Drives	38	82.58	91.47				
Stone-Steel-Towers	10	98.70	82.83				
Total/Average	1025	94.06	94.32±2.58	Average	4279	88.26	88.09±0.29

VI. DISCUSSION AND CONCLUSION

In this paper, we have proposed a new algorithm for spectral–spatial hyperspectral image classification using relational features and iteration between two classifiers. According to our knowledge, this represents an innovative contribution in the related field because of the introduction of two main elements of novelty: 1) an iterative structure between two classifiers and 2) the construction of a set of relational features that are based on the spatial structure of the image and on the labels determined at the current iterative step.

The algorithm shows higher performance compared to basic classifiers (MLR and SVM), while it is comparable (in some case better) to the spectral–spatial algorithms proposed in the literature. In detail, we found that IRMC outperforms SVM on the Indian Pines data set, while the accuracy gain for the Pavia University and Salinas data sets is less marked but still present. Results have been further illustrated on the classification maps, to describe the behavior of the algorithm and its spatial-dependent capability to achieve correct classification.

IRMC has shown to take advantage of the spatial information, but probably, the use of the windows to collect the spatial features is not optimal since the lowest accuracy has been found just in high heterogeneous areas. Concerning this aspect, it can be interesting to investigate, in future, on other methods to generate the spatial features. As an example, the homogeneity index may be computed on spatial neighbors identified by windows whose shape adaptively changes with the local texture of the area, to better account for the heterogeneity; another possibility is to use a set of windows of different shape or orientation as structuring elements to construct morphological operations (as openings or closings). The SDHI could be computed after the application of the sequence of such operators applied on the labeled maps. Such operators, in fact, have been successfully used in [46].

The several ways to build relational features are related to another problem: the hyperdimensionality of the samples. In our algorithm, the number of relational features is proportional to the number of classes and of windows used (see Section III-C). With different ways to get the spatial features, a number as high as 1000 could be easily reached. In this case, methods to reduce the features by using dimensionality reduction techniques (PCA for example) become mandatory to avoid the Hughes phenomenon since the labeled set is supposed to be limited. Another possibility, again, may be to use an ensemble of classifiers to deal with the lower ratio between the initial number of training sets and the number of features.

All such possibilities and challenges will be the matter of future investigations.

APPENDIX

A. Determination of Weights for MLR

The MLR model assigns the posterior probabilities $p(v_n \in \Pi_i|\mathbf{x})$ by means of (1). To achieve the posterior probabilities for all the samples, it is necessary to fit the observations to the model, i.e., to estimate the weights $w_{0i}, \mathbf{w}_i$. Let us redefine the weights as follows:

$$\mathbf{w}_i = [w_{0i}, \mathbf{w}_i^T]^T \tag{17}$$

with $i = 1, \ldots, K-1$. Each new array has size $M+1$ so that the total number of weights to estimate is $(M+1)(K-1)$. The conditional likelihood can now be written as

$$\mathcal{L}(\mathbf{w}_1, \ldots, \mathbf{w}_{K-1}|\mathbf{X}) = \prod_{n=1}^{N} \prod_{k=1}^{K} p_{nk}^{y_{nk}} \tag{18}$$

with p_{nk} defined in (7) and y_{nk} being the target variables, defined in (8). The negative conditional log-likelihood is then

$$\ell(\mathbf{w}_1, \ldots, \mathbf{w}_{K-1}|\mathbf{X}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk} \log p_{nk} \tag{19}$$

which is also known as the cross-entropy error function [see (13)] for the multiclass classification problem.

In the logistic regression, the maximum likelihood has no closed-form solution due to the nonlinearity of the logistic

sigmoid function. Anyway, the departure from a quadratic dependence of the log likelihood function on the parameter vector is not substantial. To be precise, the error function is concave; hence, it has a unique minimum.

This minimum can be found by an efficient algorithm based on the Newton–Raphson iterative optimization scheme [38], [47], which uses a local quadratic approximation to the log-likelihood function. Making use of simple algebraic relations, we get

$$\nabla_{\mathbf{w}_j}\ell = \frac{\partial \ell}{\partial \mathbf{w}_j} = \sum_{n=1}^{N} \mathbf{x}_n (y_{nj} - p_{nj}) \tag{20}$$

where the array of explanatory variables has been redefined as $\mathbf{x} = [1\ \mathbf{x}^T]^T$. Equation (20) is a column array of $M+1$ elements and is defined for $j = 1, \ldots, K-1$. The second derivative is

$$\nabla_{\mathbf{w}_k}\nabla_{\mathbf{w}_j}\ell = \frac{\partial^2 \ell}{\partial \mathbf{w}_k \partial \mathbf{w}_j} = -\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T (I_{kj} - p_{nj}) p_{nk} \tag{21}$$

with I_{kj} being the kjth element of the identity matrix of rank $K-1$. The previous equation can be formulated in a matrix-vector product. Let us define $\mathbf{P}^{(kj)}$ as an $[N \times N]$ diagonal matrix whose elements are $\mathbf{P}^{(kj)} = [(I_{kj} - p_{nj})p_{nk}]$, for $n = 1, \ldots, N$. Let us further define $\mathbf{\Phi}$ as the matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ of size $(M+1) \times N$. For any couple (k, j), the product in (21) can be rewritten as

$$\nabla_{\mathbf{w}_k}\nabla_{\mathbf{w}_j}\ell = -\mathbf{\Phi}\mathbf{P}^{(kj)}\mathbf{\Phi}^T. \tag{22}$$

Now, we concatenate either the columns of $\mathbf{w}_k$ in a single array $\mathbf{w}$ and of $\nabla_{\mathbf{w}_k}\ell$ in $\nabla\ell(\mathbf{w})$ to get two column vectors of size $(M+1)(K-1)$, and we define the Hessian matrix

$$\mathbf{H} = \left[\nabla_{\mathbf{w}_k}\ \nabla_{\mathbf{w}_j}\ell\right] \tag{23}$$

as the $(M+1)(K-1) \times (M+1)(K-1)$ block matrix having in position (j, k) the matrix $\nabla_{\mathbf{w}_k}\nabla_{\mathbf{w}_j}\ell$ achieved in (21) or (22). The Newton–Raphson update, for minimizing the negative conditional log-likelihood $\ell(\mathbf{w})$, takes the form

$$\mathbf{w}^{(l+1)} = \mathbf{w}^{(l)} - \mathbf{H}^{-1}\nabla\ell(\mathbf{w}) \tag{24}$$

and converges to a solution, provided that a stop criterion has been defined.

B. Regularization

Searching for the global minimum of the nonlinear function defined in (19) could get it stuck in local optima. It is for this reason that the solution is often modified to account for a *regularization* term (in order to control the overfitting) so that the total error function to be minimized takes the form

$$\ell'(\mathbf{w}) = \ell(\mathbf{w}) + \gamma\Theta(\mathbf{w}) \tag{25}$$

where γ is the regularization coefficient that controls the relative importance of the data-dependent error $\ell(\mathbf{w})$ and of the regularization term $\Theta(\mathbf{w})$. One of the most used forms of regularization is given by the sum of squares of the weight vector elements

$$\Theta(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{26}$$

as this has the advantage that the error function remains a quadratic form of $\mathbf{w}$. Using the regularization term, (20) and (21) are modified as follows:

$$\nabla_{\mathbf{w}_j}\ell' = \nabla_{\mathbf{w}_j}\ell - \gamma\mathbf{w} \tag{27}$$

$$\nabla_{\mathbf{w}_k}\nabla_{\mathbf{w}_j}\ell' = \nabla_{\mathbf{w}_k}\nabla_{\mathbf{w}_j}\ell - \gamma \tag{28}$$

while (24) remains unchanged.

REFERENCES

[1] A. F. H. Goetz, G. Vane, J. E. Solomon, and B. N. Rock, "Imaging spectrometry for Earth remote sensing," *Science*, vol. 228, no. 4704, pp. 1147–1153, Jun. 1985.

[2] R. O. Green *et al.*, "Imaging spectroscopy and the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)," *Remote Sens. Environ.*, vol. 65, no. 3, pp. 227–248, Sep. 1998.

[3] G. Shaw and D. Manolakis, "Signal processing for hyperspectral image exploitation," *IEEE Signal Process. Mag.*, vol. 19, no. 1, pp. 12–16, Jan. 2002.

[4] D. A. Landgrebe, *Signal Theory Methods in Multispectral Remote Sensing*, vol. 29. Hoboken, NJ, USA: Wiley, 2005.

[5] A. Plaza *et al.*, "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.*, vol. 113, no. S1, pp. S110–S122, Sep. 2009.

[6] A. Plaza, P. Martinez, R. Perez, and J. Plaza, "A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 3, pp. 650–663, Mar. 2004.

[7] B. Demir, C. Persello, and L. Bruzzone, "Batch-mode active-learning methods for the interactive classification of remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 3, pp. 1014–1031, Mar. 2011.

[8] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Trans. Inf. Theory*, vol. IT-14, no. 1, pp. 55–63, Jan. 1968.

[9] L. Bruzzone, M. Chi, and M. Marconcini, "A novel transductive SVM for semisupervised classification of remote-sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 11, pp. 3363–3373, Nov. 2006.

[10] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[11] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 8, pp. 1778–1790, Aug. 2004.

[12] G. Camps-Valls and L. Bruzzone, "Kernel-based methods for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 6, pp. 1351–1362, Jun. 2005.

[13] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. ICML*, 1999, pp. 200–209.

[14] A. Gammerman, V. Vovk, and V. Vapnik, "Learning by transduction," in *Proc. 14th Conf. Uncertainty Artif. Intell.*, 1998, pp. 148–155.

[15] E. Pasolli, F. Melgani, D. Tuia, F. Pacifici, and W. J. Emery, "SVM active learning approach for image classification using spatial information," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 4, pp. 2217–2233, Apr. 2014.

[16] B.-C. Kuo and D. A. Landgrebe, "Nonparametric weighted feature extraction for classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 5, pp. 1096–1105, May 2004.

[17] B. Zhang, S. Li, X. Jia, L. Gao, and M. Peng, "Adaptive Markov random field approach for classification of hyperspectral imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 5, pp. 973–977, Sep. 2011.

[18] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Spectral-spatial hyperspectral image segmentation using subspace multinomial logistic regression and Markov random fields," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 3, pp. 809–823, Mar. 2012.

[19] A. Zhang and Y. Xie, "Chaos theory-based data-mining technique for image endmember extraction: Lyapunov index and correlation dimension (L and D)," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 4, pp. 1935–1947, Apr. 2014.

[20] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, no. 6, pp. 721–741, Nov. 1984.
[21] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Neural Comput.*, vol. 13, no. 10, pp. 2173–2200, Oct. 2001.
[22] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Spectral-spatial classification of hyperspectral data using loopy belief propagation and active learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 2, pp. 844–856, Feb. 2013.
[23] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2282–2312, Jul. 2005.
[24] L. Mcdowell and D. Aha, "Semi-supervised collective classification via hybrid label regularization," in *Proc. 29th ICML*, 2012, pp. 975–982.
[25] P.-G. Jaime *et al.*, "Enhanced land use/cover classification of heterogeneous tropical landscapes using support vector machines and textural homogeneity," *Int. J. Appl. Earth Observ. Geoinf.*, vol. 23, no. 8, pp. 372–383, Aug. 2013.
[26] J. A. Benediktsson, M. Pesaresi, and K. Amason, "Classification and feature extraction for remote sensing images from urban areas based on morphological transformations," *IEEE Trans. Geosci. Remote Sens.*, vol. 41, no. 9, pp. 1940–1949, Sep. 2003.
[27] P. Sen *et al.*, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, pp. 93, 2008.
[28] M. Bilgic and L. Getoor, "Effective label acquisition for collective classification," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2008, pp. 43–51.
[29] D. Jensen, J. Neville, and B. Gallagher, "Why collective inference improves relational classification," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2004, pp. 593–598.
[30] Y. Tarabalka, J. A. Benediktsson, J. Chanussot, and J. C. Tilton, "Multiple spectral-spatial classification approach for hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 11, pp. 4122–4132, Nov. 2010.
[31] S. Xiaoxiao and L. Y. Y. Philip, "Collective prediction with latent graphs," in *Proc. 20th Int. CIKM*, 2011, pp. 1127–1136.
[32] B. Mustafa, M. Lilyana, and G. Lise, "Active learning for networked data," in *Proc. 27th ICML*, 2010, pp. 79–86.
[33] L. Qing and G. Lise, "Link-based classification," in *Proc. ICML*, 2003, pp. 496–503.
[34] R. Xiang and J. Neville, "Pseudolikelihood EM for within-network relational learning," in *Proc. 8th IEEE ICDM*, 2008, pp. 1103–1108.
[35] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE ICC Geneva*, 1993, vol. 2, pp. 1064–1070.
[36] J. P. M. Schalkwijk and K. A. Post, "Correction to 'on the error probability for a class of binary recursive feedback strategies'," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 2, pp. 284, Mar. 1974.
[37] D. J. Costello, Jr. and J. He, "On the weight distribution of some turbo-like codes," in *Codes, Curves, and Signals*, ser. The Springer International Series in Engineering and Computer Science, vol. 485, San Mateo, CA, USA: Morgan Kaufmann, 1998, pp. 155–170.
[38] C. M. Bishop *et al.*, *Pattern Recognition and Machine Learning*, vol. 1, New York, NY, USA: Springer-Verlag, 2006.
[39] A. J. Izenmann, *Modern Multivariate Statistical Techniques*, vol. 1, New York, NY, USA: Springer-Verlag, 2008.
[40] G. C. Cawley and N. L. C. Talbot, "Efficient model selection for kernel logistic regression," in *Proc. IEEE 17th ICPR*, 2004, vol. 2, pp. 439–442.
[41] I. T. Nabney, "Efficient training of RBF networks for classification," *Int. J. Neural Syst.*, vol. 14, no. 3, pp. 201–208, Jun. 2004.
[42] X. Kong, X. Shi, and S. Y. Philip, "Multi-label collective classification," in *Proc. SDM*, 2011, vol. 11, pp. 618–629.
[43] B. Krishnapuram, L. Carin, M. A. T. Figueiredo, and A. J. Hartemink, "Sparse multinomial logistic regression: Fast algorithms and generalization bounds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 957–968, Jun. 2005.
[44] I. Jolliffe, *Principal Component Analysis*. Hoboken, NJ, USA: Wiley, 2005.
[45] J. Li, P. R. Marpu, A. Plaza, J. M. Bioucas-Dias, and J. A. Benediktsson, "Generalized composite kernel framework for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 9, pp. 4816–4829, Sep. 2013.
[46] J. A. Benediktsson, J. A. Palmason, and J. R. Sveinsson, "Classification of hyperspectral data from urban areas based on extended morphological profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 3, pp. 480–491, Mar. 2005.
[47] J. R. Movellan, "Tutorial on Multinomial Logistic Regression," Jun. 2013, [Online]. Available: http://mplab.ucsd.edu/tutorials/MultivariateLogisticRegression.pdf

Pietro Guccione was born in Bari, Italy, in 1971. He received the Laurea degree (cum laude) and the Ph.D. degree in electronic engineering from the Polytechnic University of Bari, Bari, Italy, in 1996 and 2000, respectively. He is currently an Assistant Professor with the Department of Electric and Information Engineering at the Polytechnic University of Bari. He teaches graduate-level courses in Signal Theory and Digital Communication Systems. He has authored more than 60 papers in the field of synthetic aperture radar, signal processing, and multivariate statistical analysis. He has contributed and coordinated several research contracts/projects in the field of remote sensing, working with ESA, ASI, and many private international aerospace companies.

Luigi Mascolo received the B.S. degree in telecommunications engineering from the University of Basilicata, Italy, in 2008 and the M.S. degree in telecommunications engineering from the Polytechnic University of Bari, Bari, Italy, in 2011.

In January 2012, he joined the School of Doctorate in Electric and Information Engineering at the Polytechnic University of Bari. Currently, his major research interests include multivariate data analysis, signal processing, and data mining, as well as their applications to remote sensing and neuroimaging.

Annalisa Appice received the Ph.D. degree in computer science from the University of Bari "Aldo Moro," Bari, Italy, in 2005. She is currently an Assistant Professor in the Department of Informatics, University of Bari Aldo Moro. Her research activity mainly concerns data mining, particularly multirelational data mining, spatial data mining, stream data mining, sensor data analysis, and process mining. She has published more than 120 papers in international journals and conference proceedings. She is involved in many European and national projects on data mining and has served in the program committee of more than 30 international conferences and workshops of data mining. She has cochaired five international and national workshops and acted as guest editor of two special issues of international journals (topics: Progress on Multi-Relational Data Mining and Mining Complex Patterns).