Large-scale feature selection with Gaussian mixture models for the classification of high dimensional remote sensing images

Adrien Lagrange, Mathieu Fauvel, Manuel Grizonnet

Large-scale feature selection with Gaussian mixture models for the classification of high dimensional remote sensing images

Adrien Lagrange *Student, IEEE*, Mathieu Fauvel *Senior, IEEE* and Manuel Grizonnet

***Abstract*—A large scale feature selection wrapper is discussed for the classification of high dimensional remote sensing. An efficient implementation is proposed based on intrinsic properties of Gaussian mixtures models and block matrix. The criterion function is split into two parts : one that is updated to test each feature and one that needs to be updated only once per feature selection. This split saved a lot of computation for each test. The algorithm is implemented in C++ and integrated into the Orfeo Toolbox. It has been compared to other classification algorithms on two high dimension remote sensing images. Results show that the approach provides good classification accuracies with low computation time.**

I. Introduction

With the increasing number of remote sensing missions, the quantity of available Earth observation data for a given landscape becomes larger and larger. Satellite missions produce a huge amount of data on a regular (daily) basis. From 2018, the EnMAP (Environmental Mapping and Analysis Program) satellites managed by the German space agency will produce images with 244 spectral bands, a spatial resolution of 30x30m per pixel and with a frequency of revisit of 4 days [1]. The Hyperspectral Infrared Imager (HyspIRI) of NASA will also deliver images with 212 spectral bands. Additionally to hyperspectral data, the amount of available hypertemporal data increases a lot. For instance, the European satellites Sentinel-2 were launched recently and 2 Terabytes of data will be released every day [2]. The Landsat open archive (http://landsat.usgs.gov/products_data_at_no_charge.php) has also released thousands of images. Such high volume of Earth observation data provides accurate information of land state and functions, and helps to improve the understanding of the planet [3]. However, processing such data is more and more challenging because of statistical and computational issues.

In the spectral or temporal domain, a pixel is represented by a vector for which each component corresponds to a spectral/temporal measurement. The size of the vector is therefore the number of spectral or temporal measurements. For hyperspectral images, this number is typically about several hundreds while for the Sentinel-2 multitemporal images, the number of spectro-temporal measurement for a given year is approximately one thousand. When working in high dimensional spaces, statistical methods made for low or moderate dimensional spaces do not adapt well. For instance, the rate of convergence of the statistical estimation decreases when the dimension grows while jointly the number of parameters to estimate increases, making the estimation of the model parameters very difficult [4]. Consequently, with a limited training set, beyond a certain limit, the classification accuracy actually decreases as the number of features increases [5]. For the purpose of classification, these problems are related to the *curse of dimensionality* [4]. This is a major drawback in many remote sensing applications since it is difficult to collect a large and accurate ground-truth. An intensive work has been performed in the remote sensing community to build accurate classifiers for high dimensional images. Bayesian models [6], feature extraction and feature reduction techniques [6], [7], random forest [8], neural networks [9] and kernel methods [10] have been investigated for the classification of such images.

The volume of the data is increasing dramatically with respect to the number of measurement per pixel. The data volume of an hyperspectral image is typically several hundreds of Gigabytes per acquisition ($\approx 300\text{km}^2$). Multitemporal data are now available freely from internet stream (see for instance the Copernicus data hub https://cophub.copernicus.eu/). This very large volume of data requires specific computing infrastructure. High performance computing is actually investigated by the remote sensing community [11], [12]. Main issues are related to the use of parallel approaches (multi-core, GPU, clusters) to improve the processing time, and to the use of streaming techniques when data does not fit in memory. One popular open source software solution is the Orfeo Toolbox, developed by the French Space Agency (CNES) [13]. Streaming and parallel computing are conveniently proposed to users/developers through several "*ready to use*" modules.

A method to reduce both statistical and computational issues is to perform a reduction of the dimension. In fact, with the *curse of dimensionality* comes the *blessing of the dimensionality* [14]: high dimensional data spaces exhibit interesting properties for classification purpose. In particular, it is possible to get a parsimonious representation of the data while maintaining or increasing the classification accuracy [15]. For instance, in land-cover classification, given a set of spatial, temporal and spectral features, it is possible to extract those which are the most discriminant for the purpose of classification [16]. In hyperspectral data, from the hundreds of available spectral channels, it is possible to reduce the

A. Lagrange and M. Fauvel are with the Université de Toulouse, INP-ENSAT, UMR 1201 DYNAFOR, France and with the INRA, UMR 1201 DYNAFOR, France.

M. Grizonnet is with Centre National d'Études Spatiales, French Space Agency, Toulouse, France.

This work was supported by the French National Research Agency (ANR) under Project Grant ANR-13-JS02-0005-01 (Asterix project).

number of channels to make the processing more efficient in terms of statistical complexity and computational load. In short, by reducing the dimension, better classification results are expected with a reduced computational load.

There are two main strategies to reduce dimension [17, Chapter 1]: feature extraction and feature selection. Feature extraction means reformulate and summarize the information by creating new features in combining the existing ones, it is sometimes referred to as *feature construction*. Linear combination of the initial features can be extracted using Principal Component Analysis (PCA) [15] or Independent Component Analysis [18]. Supervised extraction method has also been investigated such as Fischer discriminant analysis and decision boundary feature extraction [6]. To the contrary, feature selection extracts a subset of existing features identified as the most relevant by a given criterion. This subset has the additional advantage to be much more understandable for the end-user than those constructed by a (non-)linear combination.

Feature selection/extraction algorithms can be divided into three classes. The first class is called *filter methods*. They select features independently of the classifier. Features are ranked according to some statistical measures, *e.g.*, correlation or independence. For example, PCA is a typical unsupervised filter method. Bruzzone *et al.* [19] develop a supervised filter method based on Jeffries-Matusita distance to maximize the separability of class distribution. Correlation between bands has been explored for feature selection in hyperspectral data [20]. In general, these methods are fast and do not depend on any classifier. But they do not take into account the properties of the chosen classifier and do not optimize directly the classification accuracy.

The second class are known as *wrapper methods*. They search for the best subset of variables for a given learning model. Since exhaustive searches are too expensive in terms of processing time, several sub-optimal search strategies have been designed, mainly iterative forward or backward search [21], [22] or a combination of both [23]. The advantage of such methods compared to filter methods is that they are dedicated to a particular model and to a particular learning problem. On the other hand, as they require the training of multiple models to test various set of variables, they are more time consuming.

The third class corresponds to the *embedded methods*. They do not separate the feature selection process from the learning algorithm and allow interactions between the two processes. A popular embedded method is the *Random Forest*. Embedded methods also exist for other models, *e.g.* SVM [24]–[26].

Despite a large diversity of methods, feature selection algorithms usually do not scale well with the number of pixels to be processed [27]. The training computational load is too important to compensate the reduced prediction computational load. Hence, feature selection is not widely used in operational situations. However, methods based on Gaussian Mixture Models (GMM) have several interesting properties that make them suitable for feature selection in the context of large amount of data. By taking advantage of their intrinsic properties, it is possible to increase the computational efficiency with respect to standard implementation.

The contribution of this paper is a extension of the forward feature selection method proposed in [27]. A smart implementation of the feature selection update rules are presented in order to perform efficiently on large amount of data. The rules use on linear algebra on block matrices applied to the covariance matrix of the conditional class density function. Furthermore, a floating version of the algorithm is proposed and evaluated. Several correctness of fit *criteria are proposed to handle unbalanced training sets, extending the conventional* overall accuracy *measure. Finally, the developed algorithm is made available to the scientific community as a remote module of the Orfeo Toolbox [13].*

The remaining of the article is organized as follows. Section II presents GMM classifiers and problems related to high-dimensional feature spaces. The feature selection methods are detailed in Section III. Then, an efficient implementation is presented in Section IV. Experimental results on two real high dimensional datasets are given Section VI. Conclusion and perspectives conclude the paper in Section VII.

II. Gaussian Mixture Models in high dimensional spaces

The following notations are used in the remaining. $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ denotes the training set where $\mathbf{x}_i \in \mathbb{R}^d$ is the vector of features of the i^{th} sample, d the number of spectral/temporal features, $y_i = 1, ..., C$ the associated label, C the total number of classes, n the number of samples and n_c the number of samples of class c.

A. Gaussian Mixture Models

For mixture models, it is assumed that a given sample $\mathbf{x}$ is the realization of a random vector which distribution is a mixture (convex combination) of several class conditioned distributions [28]:

$$p(\mathbf{x}) = \sum_{c=1}^{C} \pi_c f_c(\mathbf{x}|\theta) \tag{1}$$

where π_c is the prior, *i.e.*, the proportion of class c and f_c a parametric density function controlled by θ.

Among the possible parametric models, the Gaussian one is the most used [14]. It assumes that each f_c is, conditionally to c, a Gaussian distribution of parameters $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$:

$$f_c(\mathbf{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = \frac{1}{(2\pi)^{\frac{d}{2}} |\boldsymbol{\Sigma}_c|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_c)^t \boldsymbol{\Sigma}_c^{-1} (\mathbf{x} - \boldsymbol{\mu}_c)\right). \tag{2}$$

It is referred to as Gaussian mixture model (GMM). In a supervised learning framework, the class parameters $\boldsymbol{\mu}_c$, $\boldsymbol{\Sigma}_c$ and the prior π_c are usually estimated through the conventional unbiased empirical estimators:

$$\hat{\pi}_c = \frac{n_c}{n}, \tag{3}$$

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{n_c} \sum_{\{i|y_i=c\}} \mathbf{x}_i, \tag{4}$$

$$\hat{\boldsymbol{\Sigma}}_c = \frac{1}{(n_c - 1)} \sum_{\{i|y_i=c\}} (\mathbf{x}_i - \boldsymbol{\mu}_c)(\mathbf{x}_i - \boldsymbol{\mu}_c)^t. \tag{5}$$

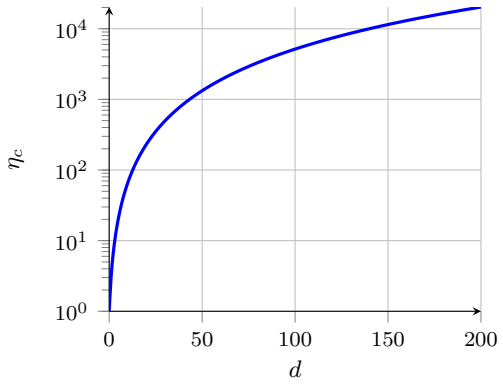


Fig. 1. Number of parameters η_c per class in function of dimension d: $\eta_c = d(d+3)/2 + 1$.

To predict the class of a new unseen sample, the maximum *a posteriori* rule is used:

$$\mathbf{x} \text{ belongs to } c \Leftrightarrow c = \arg\max_{c \in C} p(c)p(\mathbf{x}|c).$$

Under the GMM, and identifying $p(c)$ as π_c and $p(\mathbf{x}|c)$ as $f_c(\mathbf{x}|\theta)$ and by taking the log, the decision function is obtained

$$\begin{aligned} Q_c(\mathbf{x}) &= 2\log\left(p(c)p(\mathbf{x}|c)\right) \\ &= -(\mathbf{x}-\boldsymbol{\mu}_c)^t\boldsymbol{\Sigma}_c^{-1}(\mathbf{x}-\boldsymbol{\mu}_c) \\ &\quad -\log(|\boldsymbol{\Sigma}_c|) + 2\log(\pi_c) - d\log(2\pi). \end{aligned} \quad (6)$$

B. Curse of dimensionality in GMM

The computation of eq. (6) requires the inversion of the covariance matrix and the computation of the logarithm of the determinant. The estimation of these terms suffers from the curse of dimensionality [14]. In practice, the number of parameters η_c to estimate for each class increases quadratically with respect to the number of features, as illustrated in Figure 1. Hence, if the number of observation n_c is small compared to the number of parameters η_c, the estimated covariance matrix is badly conditioned and thus the computation of its inverse and its determinant would be unstable. The worst situation is $n_c < \eta_c$ which leads to a singular covariance matrix. Unfortunately, this situation happens regularly in remote sensing. For instance in hyperspectral image classification, very few labeled samples are usually available because of the difficulty and the cost to collect ground-truth.

There are two major solutions to this problem. The first option is to stabilize the inversion of the covariance matrices. Some methods investigate the use of constraints on the direct problem. Reynolds *et al.* [29] proposed to use diagonal covariance matrices. It is also possible to force the diagonal element to be higher than a given value by maximizing the GMM likelihood [30]. Celeux and Govaert [31] suggest to use equality constraints between coefficients of the covariance matrix in a parsimonious cluster-based GMM framework. Other papers propose to work on the inverse problem. A classical method is to use a regularization method as the well-known ridge regularization [32]. A ridge regularization aims to stabilize the inversion by replacing the covariance matrix $\boldsymbol{\Sigma}_c$ by $\boldsymbol{\Sigma}_c + \tau\mathbf{I}$ where τ is a positive parameter and $\mathbf{I}$ the identity matrix. Jensen *et al.* [33] propose a different approach using a sparsity approximation to inverse the covariance matrix.

The second option is to reduce the dimension. Feature extraction/selection methods have been developed in order to reduce the dimension with various approaches described in Section I. In this study, this latter option is explored and a feature selection method named sequential forward features selection is presented.

III. Sequential forward features selection

The feature selection method proposed in this work is a wrapper method associated to GMM models. Two elements are needed to set up a wrapper method:

1) A function that ranks the features according to some good classification or class separability criterion,
2) A search strategy to optimize the function.

Section III-A describes the various criteria used in this work and two search strategies are discussed in III-B.

A. Criterion function

The criterion evaluates how a given model built with a subset of features performs for the classification task. It can be an estimation of the correct classification or a measure of separability/similarity between class distributions. The former are in general more demanding in terms of processing time than the later.

1) Measures of correct classification: A measure of correct classification is based on an error matrix M, or *confusion matrix* [34, Chapter 4]. The confusion matrix allows the computation of several global and per-class indices related to the classification accuracy [34]. Three global criteria were used in this work:

- *The overall accuracy* (OA) is the rate of the number of samples with the correct predicted label over the total number of samples [34]. This metric is easy to interpret but is biased in the case of unbalanced classes.
- *The Cohen's kappa* (K) is a statistic which measures the probability of agreement between predictions and ground-truth [34].
- *The mean F1 score* (F1mean) is the average of the F1 score for each class and the F1 score is the harmonic mean of the precision (number of True Positive over True Positive plus False Positive) and the recall (number of True Positive over True Positive plus False Negative) [35].

High values of theses indices correspond to an accurate classification.

These indices are estimated from the training set by a n_{cv}-cross-validation (n_{cv}-CV) [36]. To compute the n_{cv}-CV, a subset is removed from $\mathcal{S}$ and the GMM is learned with the remaining training samples. A test error is computed with the removed training samples used as validation samples. The process is iterated n_{cv} times and the estimated classification rate is computed as the mean test error over the n_{cv} subsets of $\mathcal{S}$.

2) Similarity between distributions: The similarity between two distributions can be quantified using divergence measures [37]. Contrary to measures of correct classification, divergences are computed directly on the trained model, with no need of cross-validation estimation. Two particular divergences are used in this work: the *Kullback-Leibler* divergence and the *Jeffries-Matusita* distance. The advantage of these divergences is that they have an explicit expression in the case of Gaussian models. The simplification allows to get rid of any integration calculations which is a major problem when dealing with high-dimensional data.

The Kullback-Leibler divergence (KL divergence) measures the amount of information lost when the first distribution is approximated by the second one [38]. It can be explicitly computed in the case of Gaussian distributions:

$$KL_{cc'} = \frac{1}{2}\Bigg\{\mathrm{Tr}(\boldsymbol{\Sigma}_c^{-1}\boldsymbol{\Sigma}_{c'}) + (\boldsymbol{\mu}_c - \boldsymbol{\mu}_{c'})^t\boldsymbol{\Sigma}_c^{-1}(\boldsymbol{\mu}_c - \boldsymbol{\mu}_{c'}) - d + \log\left(\frac{|\boldsymbol{\Sigma}_c|}{|\boldsymbol{\Sigma}_{c'}|}\right)\Bigg\}, \quad (7)$$

where Tr is the trace operator and d the dimension of the distribution.

The KL divergence is not symmetric, *i.e.*, $KL_{cc'} \neq KL_{c'c}$. A symmetrical version is used to compute the criterion function:

$$\begin{aligned} SKL_{cc'} &= KL_{cc'} + KL_{c'c} \\ &= \frac{1}{2}\Bigg\{\mathrm{Tr}(\boldsymbol{\Sigma}_c^{-1}\boldsymbol{\Sigma}_{c'} + \boldsymbol{\Sigma}_{c'}^{-1}\boldsymbol{\Sigma}_c) \\ &\quad + (\boldsymbol{\mu}_c - \boldsymbol{\mu}_{c'})^t(\boldsymbol{\Sigma}_c^{-1} + \boldsymbol{\Sigma}_{c'}^{-1})(\boldsymbol{\mu}_c - \boldsymbol{\mu}_{c'}) - 2d\Bigg\}. \end{aligned} \quad (8)$$

The extension to the multiclass problem is done by taking the weighted mean of the KL divergences computed on all pair of classes [19]:

$$C_{SKL} = \sum_{c=1}^{C}\sum_{c'=c+1}^{C} \pi_c\pi_{c'} SKL_{cc'}. \quad (9)$$

The Bhattacharyya distance is defined in the case of Gaussian model as

$$\begin{aligned} B_{cc'} =& \frac{1}{8}(\boldsymbol{\mu}_c - \boldsymbol{\mu}_{c'})^t\left(\frac{\boldsymbol{\Sigma}_c + \boldsymbol{\Sigma}_{c'}}{2}\right)^{-1}(\boldsymbol{\mu}_c - \boldsymbol{\mu}_{c'}) \\ &+ \frac{1}{2}\log\left(\frac{|\boldsymbol{\Sigma}_c + \boldsymbol{\Sigma}_{c'}|}{\sqrt{|\boldsymbol{\Sigma}_c||\boldsymbol{\Sigma}_{c'}|}}\right). \end{aligned} \quad (10)$$

The *Jeffries-Matusita distance* is a measure based on the Bhattacharyya distance. It saturates when the separability between the two distributions increases [39]. The JM distance is defined as

$$JM_{cc'} = \sqrt{2\{1 - \exp(-B_{cc'})\}}. \quad (11)$$

Similar to the KL divergence, a weighted mean of the distance between two classes is computed to aggregate the measures in a single value:

$$C_{JM} = \sum_{c=1}^{C}\sum_{c'=c+1}^{C} \pi_c\pi_{c'} JM_{cc'}. \quad (12)$$

TABLE I
SUMMARY OF THE DIFFERENT CRITERION FUNCTIONS.

Criterion	Type	Complexity
Overall accuracy	Accuracy	High
Cohen's kappa	Accuracy	High
F1 mean	Accuracy	High
Kullback-Leibler divergences	Divergence	Low
Jeffries-Matusita distance	Divergence	Low

Table I summarizes the presented criterion functions and their characteristics. In the following J denotes one criterion from Table I.

B. Selection method

Two sequential search algorithms have been implemented in this work [17]: the sequential forward selection and the sequential floating forward. The later one is an extension of the former. Both select features iteratively.

1) Sequential forward features selection: The Sequential Forward Selection (SFS) starts with an empty set of selected features. At each step, the feature associated to the highest criterion function J is added to the set. This feature is definitively added to the pool of selected features and the algorithm stops when a given number of variables *maxVarNb* has been reached. The Algorithm 1 presents the process in details.

Algorithm 1 Sequential forward features selection

Require: Ω, J, maxVarNb
$\Omega = \emptyset$
$F = \{\text{all variables } f_i\}$
while card(Ω) $< maxVarNb$ **do**
  for all $f_i \in F$ **do**
    $R_i = J(\{\Omega + f_i\})$
  end for
  $j = \arg\max_i R_i$
  $\Omega = \{\Omega + f_j\}$
  $F = F \setminus f_j$
end while
return Ω

2) Sequential floating forward feature selection: The Sequential Floating Forward Selection (SFFS) [23] is based on two algorithms: the SFS described above and the Sequential Backward Selection (SBS). The SBS is the backward equivalent of SFS. The difference is that it starts with every features in the pool of selected features and tries at each step to remove the less significant one in term of the given criterion function.

The SFFS works as the SFS but between each step of the SFS algorithm, a backward selection is operated to identify the less important feature. If the criterion value is higher than the best value ever obtained with a set of same size, the identified feature is picked out. The SBS step is repeated while removing the less important feature leads to an increase of the criterion value. Then SFS is called again. The algorithm stops when a given number of features *maxVarNb* has been selected. The Algorithm 2 provides details about the process.

This SFFS algorithm evaluates more solutions than the SFS algorithm. The results are expected to be better but the trade-

off is an increased computational time which is dependent on the complexity of the dataset.

Algorithm 2 Sequential floating forward features selection

Require: J, maxVarNb
1: $\Omega = \overbrace{(\emptyset, ..., \emptyset)}^{maxVarNb}$
2: $F = \{\text{all variables } f_i\}$
3: $k = 0$
4: **while** $k <$ maxVarNb **do**
5: **for all** $f_i \in F$ **do**
6: $R_i = J(\{\Omega_k + f_i\})$
7: **end for**
8: $j = \arg\max_i R_i$
9: $k = k + 1$
10: **if** $R_j \geq J(\Omega_k)$ **then**
11: $\Omega_k = \{\Omega_{k-1} + f_j\}$
12: flag $= 1$
13: **while** $k > 2$ and flag $= 1$ **do**
14: **for all** $f_i \in \Omega_k$ **do**
15: $R_i = J(\{\Omega_k \setminus f_i\})$
16: **end for**
17: $j = \arg\max_i R_i$
18: **if** $R_j > J(\Omega_{k-1})$ **then**
19: $\Omega_{k-1} = \{\Omega_k \setminus f_j\}$
20: $k = k - 1$
21: **else**
22: flag $= 0$
23: **end if**
24: **end while**
25: **end if**
26: **end while**
27: **return** Ω_{maxVarNb}

IV. Efficient implementation

The most demanding part of the algorithm is the evaluation of the criterion for all the remaining variables (see lines 5-7 in Algorithm 1). Calculations are based on linear algebra, and the numerical complexity is on average $O(d^3)$. Furthermore, for the *accuracy*-type criterion the complexity is augmented by the cross-validation procedure.

An efficient implementation of the criterion optimization is detailed in the following. It is based on the symmetry property of the covariance matrix and block inverse formula [40]. It is shown that the criterion can be split into two parts: one that needs to be computed for each tested variable, and one that needs to be computed only once per selection step. For the cross-validation part, updates rules are given to derive sub-models without the necessity to learn a GMM models for each fold.

A. Statistical update rules

1) Update for cross validation: Based on [27], a method to accelerate the n_{cv}-fold cross-validation process in the case of criterion functions based on correct classification measures was implemented. The idea is to estimate the GMM with the whole training set once and then, instead of training models on $(n_{cv} - 1)$ folds, parameters of the complete model are used to derive those of sub-models, thus reducing the whole complexity.

Proposition 1 (Mean update for cross-validation)**.**

$$\hat{\boldsymbol{\mu}}_c^{n_c - \nu_c} = \frac{n_c \hat{\boldsymbol{\mu}}_c^{n_c} - \nu_c \hat{\boldsymbol{\mu}}_c^{\nu_c}}{n_c - \nu_c}$$

Proposition 2 (Covariance matrix update cross-validation)**.**

$$\hat{\boldsymbol{\Sigma}}_c^{n_c - \nu_c} = \frac{1}{n_c - \nu_c - 1}\Big\{(n_c - 1)\hat{\boldsymbol{\Sigma}}_c^{n_c} - (\nu_c - 1)\hat{\boldsymbol{\Sigma}}_c^{\nu_c} - \frac{n_c \nu_c}{(n_c - \nu_c)}(\hat{\boldsymbol{\mu}}_c^{\nu_c} - \hat{\boldsymbol{\mu}}_c^{n_c})(\hat{\boldsymbol{\mu}}_c^{\nu_c} - \hat{\boldsymbol{\mu}}_c^{n_c})^t\Big\}$$

where n_c is the number of samples of class c, ν_c is the number of samples of class c removed from the initial set, exponents on $\boldsymbol{\Sigma}_c$ and $\boldsymbol{\mu}_c$ denotes the set of samples used to compute them.

2) Criterion function computation: At iteration k, depending on the criterion, three terms have to be computed: the inverse of the covariance matrix, the logarithm of the determinant of the covariance matrix and the quadratic term in eq. (6). However, all these terms have already been computed for iteration $(k-1)$. By using the positive definiteness of the the covariance matrix and block formulae [41, Chapter 9.2], it is possible to factorize these terms at iteration k.

In the remaining of the paper, $\boldsymbol{\Sigma}_c^{(k-1)}$ denotes the covariance matrix of the $(k-1)^{th}$ iteration, *i.e.*, the covariance matrix of the selected features and $\boldsymbol{\Sigma}_c^{(k)}$ denotes a covariance matrix at the k^{th} iteration, *i.e.*, the covariance matrix augmented by the feature x_k. Then, since $\boldsymbol{\Sigma}_c^{(k)}$ is a positive definite symmetric matrix, the covariance matrix can be written as

$$\boldsymbol{\Sigma}_c^{(k)} = \begin{bmatrix} \boldsymbol{\Sigma}_c^{(k-1)} & \mathbf{u}_c \\ \mathbf{u}_c^t & \sigma_c^{(k)} \end{bmatrix}, \tag{13}$$

where $\sigma_c^{(k)}$ is the variance of x_c, $\mathbf{u}_c$ is the k^{th} column of the matrix without the diagonal element, *i.e.*, $\mathbf{u}_c(i) = \boldsymbol{\Sigma}_c^{(k)}(i,k)$ with $i \in [1, k-1]$. Using block matrix inverse formulae, the inverse of the covariance matrix is given by the following proposition.

Proposition 3 (Forward update rule for the inverse of the covariance matrix)**.**

$$(\boldsymbol{\Sigma}_c^{(k)})^{-1} = \begin{bmatrix} \mathbf{A}_c & \mathbf{v}_c \\ \mathbf{v}_c^t & \frac{1}{\alpha_c} \end{bmatrix} \tag{14}$$

where $\mathbf{A}_c = (\boldsymbol{\Sigma}_c^{(k-1)})^{-1} + \frac{1}{\alpha_c}(\boldsymbol{\Sigma}_c^{(k-1)})^{-1}\mathbf{u}_c\mathbf{u}_c^t(\boldsymbol{\Sigma}_c^{(k-1)})^{-1}$, $\mathbf{v}_c = -\frac{1}{\alpha_c}(\boldsymbol{\Sigma}_c^{(k-1)})^{-1}\mathbf{u}_c$ and $\alpha_c = \sigma_c^{(k)} - \mathbf{u}_c^t(\boldsymbol{\Sigma}_c^{(k-1)})^{-1}\mathbf{u}_c$. This update formulae is used to obtain all the $(\boldsymbol{\Sigma}_c^{(k)})^{-1}$ corresponding to all the possible augmented set of a given selection iteration. Similar update formulae can be written for the backward step.

Using eq. (13) and (14), it is possible to deduce the following propositions (proof are given in Appendix A).

Proposition 4 (Update rule for the quadratical term)**.**

$$(\mathbf{x}^{(k)})^t(\boldsymbol{\Sigma}_c^{(k)})^{-1}\mathbf{x}^{(k)} = \underbrace{(\mathbf{x}^{(k-1)})^t(\boldsymbol{\Sigma}_c^{(k-1)})^{-1}\mathbf{x}^{(k-1)}}_{\textit{computed once per selection step}} + \underbrace{\alpha_c(\begin{bmatrix} \mathbf{v}_c^t & \frac{1}{\alpha_c} \end{bmatrix}\mathbf{x}^{(k)})^2}_{\textit{computed for each augmented set}}. \tag{15}$$

Proposition 5 (Update rule for logdet)**.**

$$\log\left(|\boldsymbol{\Sigma}_c^{(k)}|\right) = \underbrace{\log\left(|\boldsymbol{\Sigma}_c^{(k-1)}|\right)}_{\substack{\textit{computed once} \\ \textit{per selection step}}} + \underbrace{\log \alpha_c}_{\substack{\textit{computed for} \\ \textit{each augmented set}}}. \tag{16}$$

From these update rules, it is now possible to split each criterion into two parts: one computed once per selection step and one computed for each augmented set.

Proposition 6 (Decision function (6))**.**

$$\begin{aligned} Q_c(\mathbf{x}) = &- \underbrace{(\mathbf{x}^{(k-1)} - \boldsymbol{\mu}_c^{(k-1)})^t (\boldsymbol{\Sigma}_c^{(k-1)})^{-1} (\mathbf{x}^{(k-1)} - \boldsymbol{\mu}_c^{(k-1)})}_{\textit{computed once per selection step}} \\ &- \underbrace{\log\left(|\boldsymbol{\Sigma}_c^{(k-1)}|\right) + 2\log(\pi_c) + k\log(2\pi)}_{\textit{computed once per selection step}} \\ &- \underbrace{\alpha_c \left(\begin{bmatrix} \mathbf{v}_c^t & \frac{1}{\alpha_c} \end{bmatrix} (\mathbf{x}^{(k)} - \boldsymbol{\mu}_c^{(k)}) \right)^2 - \log \alpha_c}_{\textit{computed for each augmented set}}. \end{aligned} \tag{17}$$

Proposition 7 (Kullback-Leibler divergence (8))**.**

$$\begin{aligned} SKL_{cc'} = \frac{1}{2} \Bigg\{ &\underbrace{\mathrm{Tr}\left((\boldsymbol{\Sigma}_c^{(k)})^{-1} \boldsymbol{\Sigma}_{c'}^{(k)} + (\boldsymbol{\Sigma}_{c'}^{(k)})^{-1} \boldsymbol{\Sigma}_c^{(k)} \right)}_{\textit{computed for each augmented set}} \\ &+ \underbrace{\alpha(\begin{bmatrix} \mathbf{v}_c^t & \frac{1}{\alpha_c} \end{bmatrix} (\boldsymbol{\mu}_c^{(k)} - \boldsymbol{\mu}_{c'}^{(k)}))^2}_{\textit{computed for each augmented set}} \\ &+ \underbrace{\alpha(\begin{bmatrix} \mathbf{v}_{c'}^t & \frac{1}{\alpha_{c'}} \end{bmatrix} (\boldsymbol{\mu}_c^{(k)} - \boldsymbol{\mu}_{c'}^{(k)}))^2 - 2k}_{\textit{computed for each augmented set}} \\ &+ \underbrace{(\boldsymbol{\mu}_c^{(k-1)} - \boldsymbol{\mu}_{c'}^{(k-1)})^t (\boldsymbol{\Sigma}_c^{(k-1)})^{-1} (\boldsymbol{\mu}_c^{(k-1)} - \boldsymbol{\mu}_{c'}^{(k-1)})}_{\textit{computed once per selection step}} \\ &+ \underbrace{(\boldsymbol{\mu}_c^{(k-1)} - \boldsymbol{\mu}_{c'}^{(k-1)})^t (\boldsymbol{\Sigma}_{c'}^{(k-1)})^{-1} (\boldsymbol{\mu}_c^{(k-1)} - \boldsymbol{\mu}_{c'}^{(k-1)})}_{\textit{computed once per selection step}} \Bigg\}, \end{aligned} \tag{18}$$

with $(\boldsymbol{\Sigma}_c^{(k)})^{-1}$ computed with Proposition 3.

Proposition 8 (Jeffries-Matusita distance (11))**.**

$$\begin{aligned} B_{cc'} = &\underbrace{\frac{1}{4}(\boldsymbol{\mu}_c^{(k-1)} - \boldsymbol{\mu}_{c'}^{(k-1)})^t (\tilde{\boldsymbol{\Sigma}}^{(k-1)})^{-1} (\boldsymbol{\mu}_c^{(k-1)} - \boldsymbol{\mu}_{c'}^{(k-1)})}_{\textit{computed once per selection step}} \\ &+ \underbrace{\frac{1}{2}\log\left(\frac{|\tilde{\boldsymbol{\Sigma}}^{(k-1)}|}{\sqrt{|\boldsymbol{\Sigma}_c^{(k-1)}||\boldsymbol{\Sigma}_{c'}^{(k-1)}|}} \right)}_{\textit{computed once per selection step}} \\ &+ \underbrace{\frac{1}{4}\tilde{\alpha}(\begin{bmatrix} \tilde{\mathbf{v}}^t & \frac{1}{\tilde{\alpha}} \end{bmatrix} (\boldsymbol{\mu}_c^{(k)} - \boldsymbol{\mu}_{c'}^{(k)}))^2 + \frac{1}{2}\log\left(\frac{\tilde{\alpha}}{\sqrt{\alpha_c \alpha_{c'}}} \right)}_{\textit{computed for each augmented set}}, \end{aligned} \tag{19}$$

where $\tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_c + \boldsymbol{\Sigma}_{c'}$ and $\tilde{\alpha}$ and $\tilde{\mathbf{v}}$ are defined as α_c and $\mathbf{v}_c$ but using $\tilde{\boldsymbol{\Sigma}}$ instead of $\boldsymbol{\Sigma}_c$.

The Algorithm 3 illustrates the optimization of the Algorithm 2 using these formulae.

Algorithm 3 Sequential floating forward features selection with updates

Require: J, maxVarNb
$\Omega = \overbrace{(\emptyset, ..., \emptyset)}^{maxVarNb}$
$F = \{\text{all variables } f_i\}$
$k = 0$
while $k <$ maxVarNb **do**
  for all $c \in \{1, ..., C\}$ **do**
    Diagonalize $\boldsymbol{\Sigma}_c^{(k-1)} = \mathbf{P}_c \boldsymbol{\Lambda}_c \mathbf{P}_c^t$
    for all $\lambda_c(i)$ **do** $\lambda_c(i) = \max(\text{EPS\_FLT}, \lambda_c(i))$
    Precompute $(\boldsymbol{\Sigma}_c^{(k-1)})^{-1}$, $(\mathbf{x}^{(k-1)} - \boldsymbol{\mu}_c^{(k-1)})^t (\boldsymbol{\Sigma}_c^{(k-1)})^{-1} (\mathbf{x}^{(k-1)} - \boldsymbol{\mu}_c^{(k-1)})$ and $\log\left(|\boldsymbol{\Sigma}_c^{(k-1)}|\right)$ using Propositions (3), (4) and (5)
  end for
  for all $f_i \in F$ **do**
    for all $c \in \{1, ..., C\}$ **do**
      Compute update constant α_c
      $\alpha_c = \max(\text{EPS\_FLT}, \alpha_c)$
    end for
    $R_i = J(\{\Omega_k + f_i\})$ using Equations (17), (18) or (19)
  end for
  $j = \arg\max_i R_i$
  $k = k + 1$
  if $R_j \geq J(\Omega_k)$ **then**
    $\Omega_k = \{\Omega_{k-1} + f_j\}$
    flag $= 1$
    while $k > 2$ and flag $= 1$ **do**
      for all $c \in \{1, ..., C\}$ **do**
        Diagonalize $\boldsymbol{\Sigma}_c^{(k-1)} = \mathbf{P}_c \boldsymbol{\Lambda}_c \mathbf{P}_c^t$
        for all $\lambda_c(i)$ **do** $\lambda_c(i) = \max(\text{EPS\_FLT}, \lambda_c(i))$
        Precompute $(\boldsymbol{\Sigma}_c^{(k-1)})^{-1}$, $(\mathbf{x}^{(k-1)} - \boldsymbol{\mu}_c^{(k-1)})^t (\boldsymbol{\Sigma}_c^{(k-1)})^{-1} (\mathbf{x}^{(k-1)} - \boldsymbol{\mu}_c^{(k-1)})$ and $\log\left(|\boldsymbol{\Sigma}_c^{(k-1)}|\right)$ using Propositions (3), (4) and (5)
      end for
      for all $f_i \in \Omega_k$ **do**
        for all $c \in \{1, ..., C\}$ **do**
          Compute update constant α_c
          $\alpha_c = \max(\text{EPS\_FLT}, \alpha_c)$
        end for
        $R_i = J(\{\Omega_k \setminus f_i\})$ using Equations (17), (18) or (19)
      end for
      $j = \arg\max_i R_i$
      if $R_j > J(\Omega_{k-1})$ **then**
        $\Omega_{k-1} = \{\Omega_k \setminus f_j\}$
        $k = k - 1$
      else
        flag $= 0$
      end if
    end while
  end if
end while
return Ω_{maxVarNb}

B. Numerical issues

For each iteration k, after the selection of optimal features w.r.t the selected criterion, the inverses of the covariance matrices and their log-determinant needs to be computed. However, the lack of training samples or the highly correlated features may induce a badly-conditioned matrix with very small, or even negative, eigenvalues. Such values will degrade drastically the estimation of the inverse and of the log-determinant, and so the numerical stability.

To deal with this limitation, the choice has been made to perform an eigenvalues decomposition of the covariance matrix $\boldsymbol{\Sigma}_c^{(k)}$:

$$\boldsymbol{\Sigma}_c^{(k)} = \mathbf{P}_c^{(k)} \boldsymbol{\Lambda}_c^{(k)} (\mathbf{P}_c^{(k)})^t \tag{20}$$

where $\boldsymbol{\Lambda}_c^{(k)}$ and $\mathbf{P}_c^{(k)}$ are the diagonal matrix of eigenvalues of the covariance matrix and the orthonormal matrix of corresponding eigenvectors, respectively. To prevent numerical instability, *non strictly positive eigenvalues are thresholded to*

```
otbcli_TrainGMMSelectionApp -io.il hyper.tif \
                            -io.vd reference.shp \
                            -gmm.varnb 20 -gmm.method forward -gmm.crit jm\
                            -gmm.best 1 -gmm.seed 0\
                            -io.out model.txt

otbcli_PredictGMMApp -in hyper.tif \
                     -model model.txt -modeltype selection\
                     -out ThematicMap.tif
```

Fig. 2. OTB Module: The feature selection is done on the image *hyper.tif* using the training set from *reference.shp*. The feature selection algorithm is the forward search used with the Jeffries-Matusita criterion, 20 features are extracted and the corresponding GMM model is saved in *model.txt*. Then the whole image is classified using the model and the results is saved in the geotiff *ThematicMap.tif*.

a fixed value EPS_FLT. In our implementation, EPS_FLT is set to the floating machine precision.

Then, the inverse of the covariance matrix can be computed as

$$(\mathbf{\Sigma}_c^{(k)})^{-1} = \mathbf{P}_c^{(k)}(\tilde{\mathbf{\Lambda}}_c^{(k)})^{-1}(\mathbf{P}_c^{(k)})^t \tag{21}$$

and the log-determinant as

$$\log\left(|\mathbf{\Sigma}_c^{(k)}|\right) = \sum_{i=1}^{d} \log(\tilde{\lambda}_c^{(k)}(i)), \tag{22}$$

where the $\tilde{}$ indicated thresholded values and $\lambda_c(i)$ the ith eigenvalue.

Same reasoning applied to the term α in the update rules: it is also thresholded to EPS_FLT. Algorithm 3 details when computational stability is enforced (lines 7, 13, 25 and 31).

C. Implementation

The proposed method has been implemented in C++ through the Orfeo Toolbox (OTB) [13]. The Orfeo Toolbox is an open-source library for remote sensing image processing, developed by the French Space Agency (CNES). The feature selection algorithm can be installed as a remote module, the source code is freely available[1].

Following OTB framework, two applications are available. The first, called `otbcli_TrainGMMSelectionApp`, performs a feature selection algorithm (SFS or SFFS) with the one criterion given from Table I. The second, called `otbcli_PredictGMMApp`, generates the thematic maps according to the learn model. The only technical limitation that the training set must fit in the RAM of the computer. The classification step is streamed and there is no limitation in term of image size. The Figure 2 shows a code excerpt to run the application.

V. Datasets

Numerical experiments have been conducted on two different datasets. The first one, called *Aisa*, is an airborne hyperspectral dataset and the second, called *Potsdam*, is an very high resolution multispectral airborne image.

[1]https://www.orfeo-toolbox.org/external-projects/

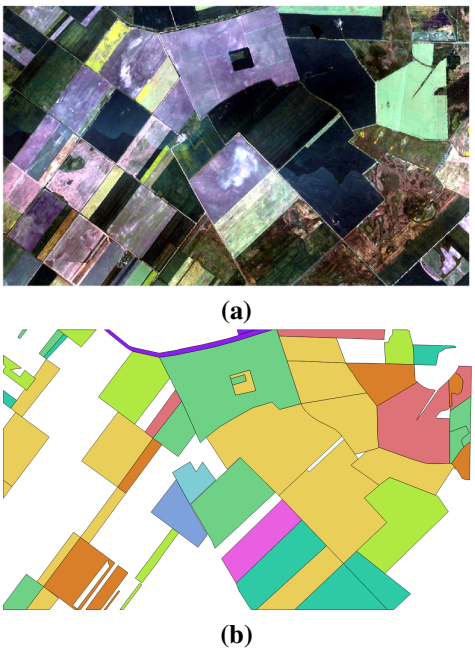


Fig. 3. Aisa dataset: **(a)** colored composition of the image (R: 634nm, G: 519nm, B: 477nm), **(b)** ground-truth.

TABLE II
Information classes for the Aisa dataset.

Class	Number of samples
Winter wheat	136,524
Sunflower	61,517
Green fallow last year treatment	30,197
Alfalfa	17,626
Maize	18,278
Millet	7,199
Broadleaved forest	10,746
Meadow	23,283
Winter barley	2,799
Reed	4,222
Water course	4,773
Rape	26,566
Green fallow with shrub	9,272
Green fallow last year treated	3,426
Pasture	2,107
Oat	3,436

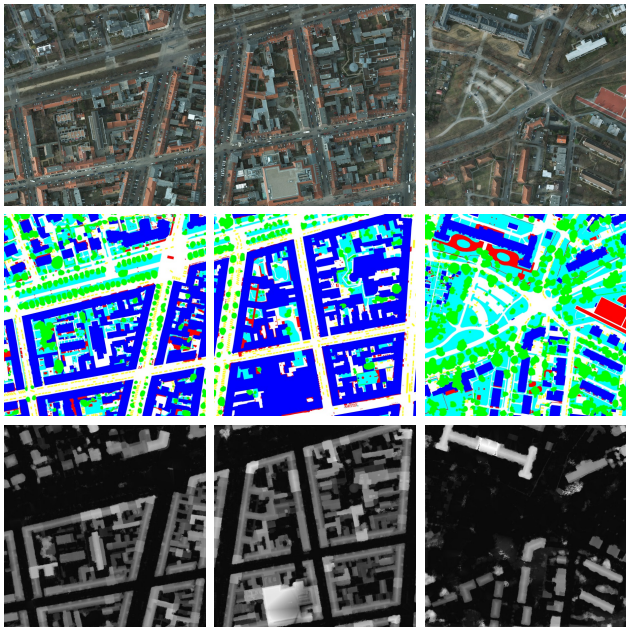
A. Aisa dataset

The Aisa dataset has been acquired by the AISA Eagle sensor during a flight campaign over Heves, Hungary. It contains 252 bands ranging from 395 to 975 nm. 16 classes have been defined for a total of 361,971 referenced pixels, Table II presents the number of pixel per class. The Figure 3 shows a colored composition of the image and the ground-truth.

B. Potsdam dataset

This second dataset is built from a dataset of remote sensing images distributed by the International Society for Photogrammetry and Remote Sensing (ISPRS)[2]. The dataset

[2]http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html



(a) **(b)** **(c)**

Fig. 4. From top to bottom, true color composition, ground-truth and normalized DSM of: **(a)** tile 5_11, **(b)** tile 5_12 and **(c)** tile 3_10.

TABLE III
INFORMATION CLASSES FOR THE THREE TILES OF THE POSTDAM DATASET.

	Number of samples per tile		
Class	5_11	5_12	3_10
Clutter	1,078,611	812,038	1,890,467
Trees	4,493,295	2,132,368	8,780,245
Cars	900,076	1,101,541	434,615
Buildings	13,469,575	17,501,421	5,128,149
Low vegetation	4,718,219	3,210,596	11,428,326
Impervious surfaces	11,340,224	11,242,036	8,338,198

is composed of aerial images of the urban area of Potsdam. The area is divided into 38 patches of 6000×6000 pixels with a resolution of 5cm by pixel and 4 channels are available: Red, Blue, Green and Infrared (RGBIR). A Digital Surface Model with the same resolution is also provided and a so-called normalized DSM representing the height above ground. The ground-truth for 24 tiles are provided with 6 classes: Low vegetation, High vegetation, Impervious surfaces, Buildings, Cars, Clutter. Three tiles have been used in this work, they are displayed in Figure 4. Table III summarizes the number of samples of each class.

In order to increase the dimensionality of the data,

Conventionally, the following features are extracted using the RGBIR images in order to increase the classification accuracy, similarly to [26]:

- Fifteen Radiometric indexes: NDVI, TNDVI, RVI, SAVI, TSAVI, MSAVI, MSAVI2, GEMI, IPVI, NDWI2, NDTI, RI, CI, BI, BI2 [42].
- Morphological profile build on each band with a disk of radius 5, 9, 13, 17, 21, 25, 29, 33, 37 and 41 (80 features) [43];
- Attribute profile build on each band with area as attribute and 1000, 2000, 5000, 10000 and 15000 as thresholds (40 features) [44].
- Attribute profile build on each band with diagonal of bounding box as attribute and 100, 200, 500, 1000 and 20000 as thresholds (40 features) [44].
- Textural features for each channel with neighborhood of 19x19 pixels: mean, standard deviation, range and entropy (16 features) [42].

The normalized DSM and the raw RGBIR image are added to these 191 features and then stacked to create a new image with 196 bands. *The resulting data cube is therefore high-dimensional.*

VI. EXPERIMENTAL RESULTS

A. Method

The aim of the experiments is to compare the proposed method to standard classifiers used in operational land map production [45]. A non-optimized previous version of the method has been already compared to other selection methods in [27]. Hence, the primary objective is to assess the operational efficiency and it is compared to other OTB classifiers used operationally through their command line applications[3].

The following classifiers are tested:

- A k-nearest-neighbors classifier (KNN) with OTB default parameters (32 as number of neighbors).
- A Random Forest classifier with parameters optimized by grid search (200 trees, 40 as max depth, 50 as size of the randomly selected subset of features at each tree node)
- A GMM classifier with ridge regularization (GMM ridge) with regularization constant optimized by grid search.

The GMM classifier is part of the external module described in Section IV-C.

All these classifiers are compared with 3 configurations of the proposed GMM classifier:

- One with forward selection and JM distance as criterion (GMM SFS JM);
- One with forward selection and Cohen's kappa as criterion (GMM SFS kappa);
- One with floating forward selection and JM distance as criterion (GMM SFFS JM).

Other configurations have been investigated and performs either equally or lower in terms of classification accuracy [46]. For the sake of clarity, only the three aforementioned configurations are discussed here *and the results for all other configurations is available in the supplementary material*.

The training set has been created with an equal number of samples for each class and additionally a spatial stratification has been performed, *i.e.*, each training sample belongs to a spatial polygon that does not intersect spatially with any spatial polygons used for the validation. Several size of training set have been tested. For the Aisa dataset, experiments have been conducted using 250, 500 and 1000 samples by class and for the Potsdam dataset, 1000 and 50000 samples by class.

[3]http://otbcb.readthedocs.io/en/latest/OTB-Applications.html

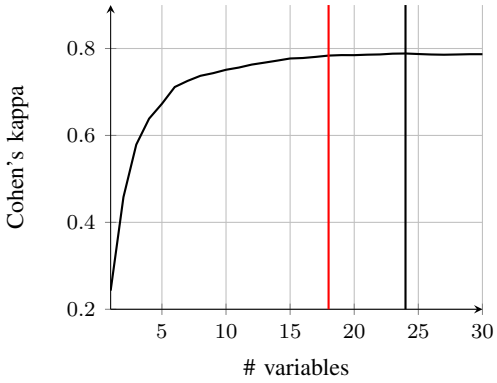


Fig. 5. Criterion evolution (kappa) in function of the number of selected variables for first trial with Aisa dataset with 500 samples by class. Red vertical line is the retained number of variables and black vertical line is the maximum of the criterion.

For SFS and SFFS selection, the number of variables to select is set to 30 for the Aisa dataset and 60 for the Potsdam dataset. After the selection procedure, the optimal number of extracted variables is selected as follow. Rather than selecting the number of variables corresponding to the highest value of the criterion, the number of retained variables is set when the criterion stops to increase significantly. *It is found by computing the discrete derivative of the criteria between two iterations and normalizing it by its maximum value. The number of selected features corresponds to the last iteration before the value drops below* 10^{-3} *for all datasets*. See Figure 5 for an example.

The classification rate is presented using Cohen's kappa *but scores computed with overall accuracy and mean of f1-score are available in supplementary material*. Processing time has been evaluated on a desktop computer with 8Gb of RAM and Intel(R) Core(TM) i5-3570 CPU @ 3.40GHz $\times$ 4 processors.

B. Aisa dataset

When creating training and validation sets, special care is taken to assure that training samples are picked out from distinct areas than test samples. The polygons of the reference are split in smaller polygons and then 50% of the polygons are taken randomly for training and the remaining 50% for validation. An example of training and validation set is shown in Figure 6. From the training polygons, a given number of samples were selected to build the training set, while all the pixels from the validation polygons were used for the validation. Moreover 20 random trials were run with a different training set (different polygons). Table IV presents the results of the experiment with mean and standard deviation of the Kappa coefficient over the 20 trials and Table V the corresponding processing time. Bold values corresponds to best results. In Table IV, when several bold scores appears for the same experiment, it means that the scores has been assessed as equivalent with a Wilcoxon rank-sum test [47]. Additionally, Figure 7 summarizes the mean of the number of selected variables for each variation of the GMM classifier with selection.

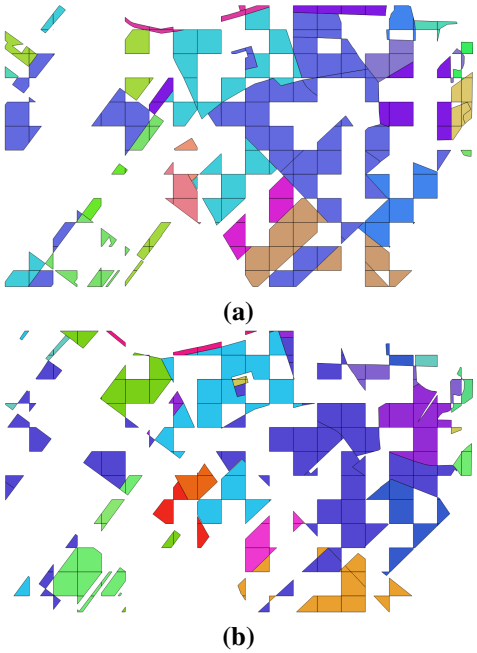


Fig. 6. Aisa dataset: **(a)** training polygons of first trial, **(b)** test polygons of first trial.

TABLE IV
AVERAGE CLASSIFICATION ACCURACY 20 TRIALS (STANDARD DEVIATION IN PARENTHESIS).

	Cohen's kappa		
# samples by class	250	500	1000
GMM SFS kappa	**0.678 (0.029)**	**0.687 (0.029)**	**0.699 (0.028)**
GMM SFS JM	**0.685 (0.030)**	**0.689 (0.030)**	**0.701 (0.029)**
GMM SFFS JM	**0.685 (0.030)**	**0.689 (0.030)**	**0.701 (0.029)**
GMM ridge	0.611 (0.040)	0.620 (0.036)	0.642 (0.034)
KNN	0.551 (0.035)	0.563 (0.033)	0.574 (0.030)
Random Forest	0.645 (0.026)	0.673 (0.023)	**0.693 (0.023)**

The results show that, on this dataset, GMM classifiers with feature selection get the best classification rate. Among the three variations of the selection algorithm, none appears to perform better than the others. Using kappa or Jeffries-Matusita distance as criterion is equal and using SFFS does not give any advantage.

The difference with the second best classifier, Random Forest, appears to be significant when using 250 and 500 samples. Random Forest has similar performance in term of classification rate with 1000 samples *and one could expect to get a better classification rate with RF if more samples were available*. The GMM classifier with ridge regularization and the KNN classifier are both outperformed.

In term of computational time, the GMM classifiers are as expected very fast for classification and also for training, except when the criterion function is a classification rate. In this case, using JM distance as criterion and SFS as search strategy is the best choice in term of time efficiency. The good performance in time can be explained by the dimension reduction. Actually, the decision rule corresponding to Equation (6) has a complexity in d^3 where d is the dimension. Thus, reducing d induces a reduction of the classification time.

TABLE V
MEAN PROCESSING TIME FOR TRAINING AND CLASSIFICATION FOR RESULTS IN TABLE IV.

	Training time (s)			**Classification time (s)**			**# of selected features**		
# samples by class	250	500	1000	250	500	1000	250	500	1000
GMM SFS kappa	257	496	955	**5.2**	**5.2**	**5.5**	11.95	12	12.05
GMM SFS JM	**8.6**	**8.9**	**9.1**	**5.7**	**5.7**	**5.9**	11.95	12	12.05
GMM SFFS JM	**8.8**	**9.0**	**9.3**	**5.0**	**5.0**	**5.4**	21.45	24.35	27.05
GMM ridge	71.7	105	167	530	530	530	all	all	all
KNN	**8.9**	19.6	59.7	387	639	887	all	all	all
Random Forest	24.5	49.3	105	33.0	41.7	45.9	all	all	all

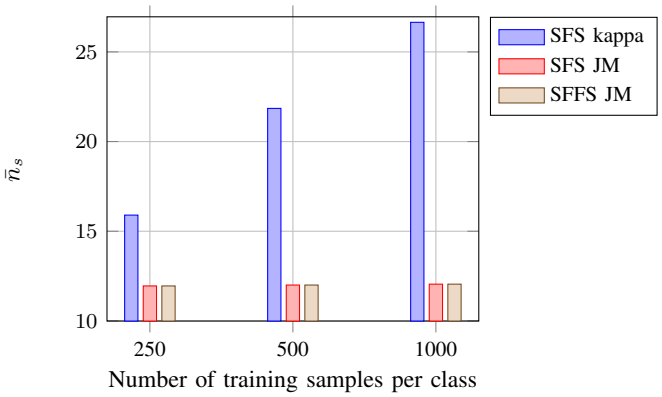


Fig. 7. Mean number $\bar{n}_s$ of selected features for the different variation of selection methods for Aisa data set. The original number of features is 252.

The processing times of the three standard classifiers suffer from the increase of training samples. For the GMM classifier with ridge, the selection of regularization parameter is more costly with more samples because of the classification rate estimation needed. For the KNN classifier, the model stores all the training samples and the prediction implies to compute the distance to all the training samples which explains the increase of the processing time and additionally of the size of the model file. Finally, for the Random Forest classifier, the trees tends to be deeper in order to capture the additional information available with more samples and that explains the increase of the processing time.

C. Potsdam dataset

For the Potsdam dataset, training samples were selected from one tile (5_11) and validation samples were all the pixel of tile 5_12 or 3_10. Table VI and Table VII present the results in terms of classification accuracy and processing time. Bold values corresponds to best results. In Table VI, when several bold scores appears for the same experiment, it means that the scores has been assessed as equivalent with a Wilcoxon rank-sum test [47].

With this second dataset, the Random Forest classifier and the GMM classifier with kappa as selection criterion perform the best in terms of classification accuracy. When using 1000 samples per class, no significant difference of classification rate has been observed on test set. But, with 50,000 samples per class, the Random Forest classifier becomes significantly better in terms of classification accuracy.

The Postdam classes are more difficult to discriminate, since Gaussianity assumption does not hold. For instance, a building can be made of various materials, resulting in heterogeneous distribution. Hence, GMM with ridge regularization performs baldy. Random Forest classifier is more adapted to this problem and reached the best classification accuracy. However, it can be note that letting the algorithm be driven by a classification quality criterion such as the Kappa coefficient helps in improving the classification accuracy. The KNN classifier is again outperformed.

From the tables VI and Table VII, the number of extracted variables shows that JM criterion identifies less relevant samples than with the kappa criterion. Moreover, the selection method with criterion kappa manages to get good performance with only 6.7% of the initial variables with 1000 samples and 15% with 50,000 samples.

In term of processing time, results are similar than with the Aisa dataset. GMM classifiers with selection are very fast for prediction. For example, the GMM classifier with kappa as criterion for the selection is 63% faster than the Random Forest classifier for prediction with 1000 samples and 83% faster with 50,000 samples. However, the training time is increased with respect to random forest.

VII. CONCLUSION AND PERSPECTIVES

An algorithm for the classification of high dimensional Earth observation images has been proposed. The algorithm is based on Gaussian mixture model and a forward feature selection strategy to reduce the dimension of the data to be processed. From experimental results, this strategy has shown to be robust the *curse of dimensionality*. As a side effect, the volume of data is reduced and the final classification processing time is reduced.

To cope with the large volume of data during the learning step, updates rules from the forward search have been split into two parts in order to save computation. One part is only computed once per iteration, and the other part needs to be computed for each tested features. Several criteria have been included, three based on classification accuracy and two based on divergence measures.

Experiments have been conducted on two real high dimensional data set, and the results have been compared to standards classifiers. Results show that the proposed approach performs, in most cases, at least as best as classifiers (Random Forest) and even outperforms all of them in term of classification time.

TABLE VI
KAPPA COEFFICIENT AND PROCESSING TIME FOR 1,000 SAMPLES BY CLASS AND AVERAGED OVER 5 TRIALS (STANDARD DEVIATION IN PARENTHESIS). PROCESSING TIMES ARE GIVEN IN SECOND.

	5_11 (train)	**5_12 (test)**	**3_10 (test)**	**Train. Time**	**Classif. time**	**# of selected features**
GMM SFS kappa	0.694 (0.002)	**0.669 (0.005)**	**0.533 (0.008)**	400	310	13.2
GMM SFS JM	0.624 (0.028)	0.631 (0.034)	0.461 (0.027)	2	310	11
GMM SFFS JM	0.624 (0.028)	0.631 (0.034)	0.461 (0.027)	2.6	310	11
GMM ridge	0.632 (0.007)	0.592 (0.010)	0.433 (0.008)	10	2000	all
KNN	0.637 (0.005)	0.607 (0.005)	0.478 (0.002)	0.7	9500	all
Random Forest	**0.729 (0.004)**	**0.673 (0.005)**	**0.529 (0.009)**	20	840	all

TABLE VII
KAPPA COEFFICIENT AND PROCESSING TIME FOR 50,000 SAMPLES BY CLASS AND AVERAGED OVER 5 TRIALS (STANDARD DEVIATION IN PARENTHESIS). PROCESSING TIMES ARE GIVEN IN SECOND. NB: THE TEST HAS NOT BEEN CONDUCT WITH KNN BECAUSE OF A TOO LONG PROCESSING TIME FOR CLASSIFICATION.

	5_11 (train)	**5_12 (test)**	**3_10 (test)**	**Train. Time**	**Classif. time**	**# of selected features**
GMM SFS kappa	0.713 (0.001)	0.684 (0.001)	0.531 (0.005)	20000	340	29
GMM SFS JM	0.560 (0.111)	0.576 (0.104)	0.435 (0.085)	6	330	10
GMM SFFS JM	0.560 (0.111)	0.576 (0.104)	0.435 (0.085)	6.6	340	10
GMM ridge	0.641 (0.015)	0.611 (0.026)	0.440 (0.015)	460	2000	all
KNN	/	/	/	/	/	/
Random Forest	**0.851 (0.001)**	**0.715 (0.001)**	**0.573 (0.002)**	2000	2000	all

The resulting code is available as a remote module of the Orfeo ToolBox on GitHub and makes it possible to process large high dimensional images efficiently. The C++ code is freely available for download: https://www.orfeo-toolbox.org/external-projects/.

Perspectives of this work concern the selection of continuous interval of features rather than a single feature [48], [49]. It will be of highest interest for continuous features, such as temporal feature or spectral feature.

APPENDIX A
PROOF OF UPDATE RULES

Proof of proposition (4).

$$
\begin{aligned}
&(\mathbf{x}^{(k)})^t(\mathbf{\Sigma}_c^{(k)})^{-1}\mathbf{x}^{(k)} \\
&= \begin{bmatrix} (\mathbf{x}^{(k-1)})^t & x_k \end{bmatrix} \begin{bmatrix} \mathbf{A}_c & \mathbf{v}_c \\ \mathbf{v}_c^t & \frac{1}{\alpha_c} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(k-1)} \\ x_k \end{bmatrix} \\
&= \begin{bmatrix} (\mathbf{x}^{(k-1)})^t & x_k \end{bmatrix} \begin{bmatrix} \mathbf{A}_c\mathbf{x}^{(k-1)} + x_k\mathbf{v}_c \\ \mathbf{v}_c^t\mathbf{x}^{(k-1)} + \frac{x_k}{\alpha_c} \end{bmatrix} \\
&= (\mathbf{x}^{(k-1)})^t\mathbf{A}_c\mathbf{x}^{(k-1)} + x_k\mathbf{v}_c^t\mathbf{x}^{(k-1)} \\
&\quad + (\mathbf{x}^{(k-1)})^t\mathbf{v}_c x_k + \frac{(x_k)^2}{\alpha_c} \\
&= (\mathbf{x}^{(k-1)})^t\Big((\mathbf{\Sigma}_c^{(k-1)})^{-1} \\
&\quad + \frac{1}{\alpha_c}(\mathbf{\Sigma}_c^{(k-1)})^{-1}\mathbf{u}_c\mathbf{u}_c^t(\mathbf{\Sigma}_c^{(k-1)})^{-1}\Big)\mathbf{x}^{(k-1)} \\
&\quad + 2x_k\mathbf{v}_c^t\mathbf{x}^{(k-1)} + \frac{(x_k)^2}{\alpha_c} \\
&= (\mathbf{x}^{(k-1)})^t\Big((\mathbf{\Sigma}_c^{(k-1)})^{-1} + \alpha_c\mathbf{v}_c\mathbf{v}_c^t\Big)\mathbf{x}^{(k-1)} \\
&\quad + 2x_k\mathbf{v}_c^t\mathbf{x}^{(k-1)} + \frac{(x_k)^2}{\alpha_c} \\
&= (\mathbf{x}^{(k-1)})^t(\mathbf{\Sigma}_c^{(k-1)})^{-1}\mathbf{x}^{(k-1)} + \alpha_c\Big((\mathbf{x}^{(k-1)})^t\mathbf{v}_c\mathbf{v}_c^t\mathbf{x}^{(k-1)} \\
&\quad + 2\frac{x_k}{\alpha_c}\mathbf{v}_c^t\mathbf{x}^{(k-1)} + \frac{(x_k)^2}{\alpha_c^2}\Big) \\
&= (\mathbf{x}^{(k-1)})^t(\mathbf{\Sigma}_c^{(k-1)})^{-1}\mathbf{x}^{(k-1)} + \alpha_c\Big((\mathbf{x}^{(k-1)})^t\mathbf{v}_c + \frac{x_k}{\alpha_c}\Big)^2 \\
&= (\mathbf{x}^{(k-1)})^t(\mathbf{\Sigma}_c^{(k-1)})^{-1}\mathbf{x}^{(k-1)} + \alpha_c(\begin{bmatrix} \mathbf{v}_c^t & \frac{1}{\alpha_c} \end{bmatrix}\mathbf{x}^{(k)})^2
\end{aligned}
$$

□

Proof of proposition (5). From eq. (13) and standard results for the determinant of block matrix [40, Chapter 9] we have

immediately:

$$\log\left(|\mathbf{\Sigma}_c^{(k)}|\right) = \log\left(|\mathbf{\Sigma}_c^{(k-1)}|\right)\log\left(\sigma_c^{(k)} - \mathbf{u}_c^t(\mathbf{\Sigma}_c^{(k-1)})^{-1}\mathbf{u}_c\right)$$
$$= \log\left(|\mathbf{\Sigma}_c^{(k-1)}|\right)\log\left(\alpha_c\right)$$

□

REFERENCES

[1] R. Mller, M. Bachmann, C. Makasy, A. D. Miguel, A. Mller, A. Neumann, G. Palubinskas, R. Richter, M. Schneider, T. Walzel, H. Kaufmann, L. Guanter, K. Segl, and D. G. Gfz, "Enmap - the future hyperspectral satellite mission product generation," in *Mller and U. Srgel (eds), ISPRS Hannover Workshop 2009, HighResolution Earth Imaging for Geospatial Information, XXXVIII1-4-7 / W5*, 2009.

[2] M. Drusch, U. Del Bello, S. Carlier, O. Colin, V. Fernandez, F. Gascon, B. Hoersch, C. Isola, P. Laberinti, P. Martimort *et al.*, "Sentinel-2: Esa's optical high-resolution mission for gmes operational services," *Remote Sensing of Environment*, vol. 120, pp. 25–36, 2012.

[3] W. Wagner, "A better understanding of our earth through remote sensing," *Remote Sensing*, vol. 1, no. 1, p. 1, 2009.

[4] D. Donoho, "High-dimensional data analysis: the curses and blessing of dimensionality," in *AMS Mathematical challenges of the 21st century*, 2000.

[5] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Trans. Inf. Theory*, vol. IT-14, pp. 55–63, January 1968.

[6] D. Landgrebe, *Signal Theory Methods in Multispectral Remote Sensing*. New Jersey: John Wiley and Sons, 2003.

[7] C. J. C. Burges, "Dimension reduction: A guided tour," *Foundations and Trends in Machine Learning*, vol. 2, no. 4, pp. 275–365, 2010.

[8] J. Ham, Y. Chen, M. Crawford, and J. Ghosh, "Investigation of the random forest framework for classification of hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 3, pp. 492 – 501, Mar. 2005.

[9] F. Ratle, G. Camps-Valls, and J. Weston, "Semisupervised neural networks for efficient hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 5, pp. 2271 –2282, May 2010.

[10] G. Camps-Valls and L. Bruzzone, Eds., *Kernel Methods for Remote Sensing Data Analysis*. Wiley, 2009.

[11] E. Christophe, J. Michel, and J. Inglada, "Remote sensing processing: From multicore to GPU," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 4, no. 3, pp. 643–652, 2011.

[12] A. Plaza, Q. Du, Y.-L. Chang, and R. L. King, "High performance computing for hyperspectral remote sensing," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 4, no. 3, pp. 528–544, 2011.

[13] E. Christophe, J. Inglada, and A. Giros, "Orfeo toolbox: a complete solution for mapping from high resolution satellite images," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, pp. 1263–1268, 2008.

[14] C. Bouveyron and C. Brunet-Saumard, "Model-based clustering of high-dimensional data: A review," *Computational Statistics & Data Analysis*, vol. 71, pp. 52–78, 2014.

[15] L. O. Jimenez and D. A. Landgrebe, "Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 28, no. 1, pp. 39–54, 1998.

[16] F. E. Fassnacht, C. Neumann, M. Förster, H. Buddenbaum, A. Ghosh, A. Clasen, P. K. Joshi, and B. Koch, "Comparison of feature reduction algorithms for classifying tree species with hyperspectral data on three central european test sites," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2547–2561, 2014.

[17] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[18] A. Villa, J. A. Benediktsson, J. Chanussot, and C. Jutten, "Hyperspectral image classification with independent component discriminant analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 12, pp. 4865–4876, 2011.

[19] L. Bruzzone, F. Roli, and S. B. Serpico, "An extension of the jeffreys-matusita distance to multiclass cases for feature selection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 33, no. 6, pp. 1318–1321, 1995.

[20] B. Demir and S. Ertürk, "Phase correlation based redundancy removal in feature weighting band selection for hyperspectral images," *International journal of Remote sensing*, vol. 29, no. 6, pp. 1801–1807, 2008.

[21] A. W. Whitney, "A direct method of nonparametric measurement selection," *IEEE Transactions on Computers*, vol. 100, no. 9, pp. 1100–1103, 1971.

[22] T. Marill and D. Green, "On the effectiveness of receptors in recognition systems," *IEEE transactions on Information Theory*, vol. 9, no. 1, pp. 11–17, 1963.

[23] P. Somol, P. Pudil, J. Novovičová, and P. Paclık, "Adaptive floating search methods in feature selection," *Pattern recognition letters*, vol. 20, no. 11, pp. 1157–1163, 1999.

[24] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[25] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping, "Use of the zero-norm with linear models and kernel methods," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1439–1461, 2003.

[26] D. Tuia, R. Flamary, and N. Courty, "Multiclass feature learning for hyperspectral image classification: Sparse and hierarchical solutions," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 105, pp. 272–285, 2015.

[27] M. Fauvel, C. Dechesne, A. Zullo, and F. Ferraty, "Fast forward feature selection of hyperspectral images for classification with gaussian mixture models," *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, vol. 8, no. 6, pp. 2824–2831, 2015.

[28] C. Fraley and A. E. Raftery, "Model-based clustering, discriminant analysis, and density estimation," *Journal of the American Statistical Association*, vol. 97, pp. 611–631, 2000.

[29] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *IEEE transactions on speech and audio processing*, vol. 3, no. 1, pp. 72–83, 1995.

[30] R. J. Hathaway, "A constrained formulation of maximum-likelihood estimation for normal mixture distributions," *The Annals of Statistics*, pp. 795–800, 1985.

[31] G. Celeux and G. Govaert, "Gaussian parsimonious clustering models," *Pattern recognition*, vol. 28, no. 5, pp. 781–793, 1995.

[32] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[33] A. C. Jensen, A. Berge, and A. S. Solberg, "Regression approaches to small sample inverse covariance matrix estimation for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 10, pp. 2814–2822, 2008.

[34] R. G. Congalton and K. Green, *Assessing the accuracy of remotely sensed data: principles PCAd practices*. CRC press, 2008.

[35] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, 2011.

[36] T. J. Hastie, R. J. Tibshirani, and J. H. Friedman, *The elements of statistical learning : data mining, inference, and prediction*, ser. Springer series in statistics. Springer, 2009.

[37] S.-i. Amari, H. Nagaoka, and D. Harada, *Methods of information geometry*, ser. Translations of mathematical monographs. Providence, R.I. American Mathematical Society Oxford (Oxford University Press), 2000.

[38] S. Kullback, "Letter to the editor: The kullback-leibler distance," *The American Statistician*, 1987.

[39] L. Bruzzone and C. Persello, "A novel approach to the selection of spatially invariant features for the classification of hyperspectral images with improved generalization capability," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 47, no. 9, pp. 3180–3191, 2009.

[40] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," nov 2012, version 20121115. [Online]. Available: http://www2.imm.dtu.dk/pubdb/p.php?3274

[41] A. R. Webb, *Statistical pattern recognition*. John Wiley & Sons, 2003.

[42] "Orfeo toolbox," https://www.orfeo-toolbox.org/Applications/RadiometricIndices.html.

[43] M. Fauvel, Y. Tarabalka, J. A. Benediktsson, J. Chanussot, and J. C. Tilton, "Advances in spectral-spatial classification of hyperspectral images," *Proceedings of the IEEE*, vol. 101, no. 3, pp. 652–675, 2013.

[44] M. Dalla Mura, J. A. Benediktsson, B. Waske, and L. Bruzzone, "Morphological attribute profiles for the analysis of very high resolution images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 10, pp. 3747–3762, 2010.

[45] J. Inglada, M. Arias, B. Tardy, O. Hagolle, S. Valero, D. Morin, G. Dedieu, G. Sepulcre, S. Bontemps, P. Defourny, and B. Koetz, "Assessment of an operational system for crop type map production using high temporal and spatial resolution satellite optical imagery," *Remote Sensing*, vol. 7, no. 9, p. 12356, 2015. [Online]. Available: http://www.mdpi.com/2072-4292/7/9/12356

[46] "Operational feature selection in gaussian mixture models," https://github.com/Laadr/report-features-selection/blob/master/report.pdf.

[47] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[48] S. B. Serpico and G. Moser, "Extraction of spectral channels from hyperspectral images for classification purposes," *IEEE transactions on geoscience and remote sensing*, vol. 45, no. 2, pp. 484–495, 2007.

[49] S. D. Backer, P. Kempeneers, W. Debruyn, and P. Scheunders, "A band selection technique for spectral classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 2, no. 3, pp. 319–323, July 2005.