Operational Feature Selection in Gaussian Mixture Models

Adrien Lagrange, Mathieu Fauvel and Manuel Grizonnet

Abstract

This article presents a forward feature selection algorithm based on Gaussian mixture model (GMM) classifiers. The algorithm selects iteratively features that maximize a criterion function which can be either a classification rate or a measure of divergence. Several variations of this algorithm are explored by changing the criterion function and also by testing a floating forward variation allowing backward step to discard already selected features.

An important effort is made in exploiting GMM properties to implement a fast algorithm. In particular, update rules of the GMM model are used to compute the criterion function with various sets of features. The result is a C++ remote module for the remote sensing processing toolbox Orfeo (OTB) developed by CNES.

Finally, the method is tested and also compared to other classifiers using two different datasets, one of hyperspectral images with a lot of spectral variables and one with heterogeneous spatial features. The results validate the fact that the method performs well in terms of processing time and classification accuracy in comparison to the standard classifiers available in OTB.

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

M. Grizonnet is with Centre Nat. d'Etudes Spatiales, French Space Agency, Toulouse, France.

Operational Feature Selection in Gaussian Mixture Models

I. Introduction

With the increasing number of remote sensing missions, the quantity of data available for a given landscape becomes larger and larger. Several missions are about to produce huge amount of data or have already produced it. After 2018, the EnMAP (Environmental Mapping and Analysis Program) satellites missioned by the German space agency will produce images with 244 spectral bands with a resolution of 30x30m and revisit every 4 days [1]. The Hyperspectral Infrared Imager (HyspIRI) of NASA will also take images with 212 spectral bands every 5 days. Additionally to hyperspectral data, hypertemporal data are also developping. The European satellites Sentinel-2 were launched successfully recently and the hypertemporal data produced by this mission will be fully available at the end of 2016 [2].

However, processing this data is more and more challenging because of statistical and computational issues. This statistical issues are often referred as the *curse of dimensionality*. The Hughes phenomenon [3] states that with a given number of samples, prediction accuracy will decays when the complexity is higher than some optimum value. The problem is the fast increase of the number of parameters to estimate in order to build a model when dimension expands [4]. Thus, a important number of labeled samples is needed to perform learning. For example, in the case of Gaussian Mixture Models, the number of parameters progresses quadratically with the dimension, e.g. with 200 describing variables, the model estimation requires at least around 20,000 samples by class.

The computational issues are multiple. The computing infrastructure needed to process data is more and more expensive because the processing might requires a GPU and a large amount of RAM to load images which can weight a dozen of gigabytes [5] [6]. The processing time is also limiting and requires to use parallelize computing (GPU, multi-threading).

A possible method to solve these issues is to perform a reduction of dimension. It is possible to extract a set of relevant features to get a parsimonious representation of the data [7].

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 [8]. In hyperspectral data analysis from the hundreds of available spectral channels, it is possible to reduce the number of channels to make the processing more efficient in terms of statistical complexity because of the reduction of the number of parameters to estimate and thus also in term of computational time. Moreover, dimensional reduction might improves the capacity of generalization of the classifier and avoid overfitting.

There are two ways to reduce dimension [9]: features extraction and feature selection. Feature extraction means creating new features by combining the existing ones, for example linear combination as in Principal Component Analysis [7]. To the contrary, features selection selects a subset of existing features. It has the advantage to be much more interpretable for the end-user. The selected subset of features corresponds to the most important features for the given task.

There is a large diversity of methods for feature selection. However, they usually do not scale well with the number of pixels to be processed [10]. Nevertheless, 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.

Several strategy have been explored to perform feature selection and, among them, wrapper methods receive a certain interest of the community. Wrapper methods can be seen as a search method to determine the best subset of variables for a given learning model. As exhaustive searches are out of question in a practical amount of time, numerous search strategies have been designed. Some optimal are under particular hypothesis [11] and other suboptimal but easier to set up [12] [13]. Such methods have the advantage to tune the selection to make the most of a particular classifier but require the training of multiple models to test various set of variables which can make them slow if the training is not optimized.

This work proposes to develop a forward feature selection method using GMM in continuation of [10] and an upgraded floating forward method. The basic method selects iteratively the meaningful features. At each step, the pool of selected

features is used to train a GMM which allows to compute a criterion function then used to rank features at the next iteration. The upgraded method introduces possible backward steps after each addition of a feature. An efficient implementation of the method is presented in order to handle large amount of data. Moreover, the developed algorithm is made available as a remote module of the C++ Orfeo Toolbox [14]. Finally, tests are conducted to compare different variations of the method and to compare also to other classifiers (GMM, Random Forest, k-nearest-neighbor) available in the Orfeo Toolbox.

The remaining of this article is organized as follows. Section II presents Gaussian Mixture Model classifiers and the features selection methods used to make them suitable for high-dimension space. The work done to develop a smart implementation of the proposed selection method is presented in Section III. And finally, the tests conducted to explore variations of the algorithm and to compare it to other standard classifiers are detailed in Section IV.

II. Gaussian Mixture Models in high dimension space

In the remaining of the paper, the following notations are used. $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ denotes the training set where $\mathbf{x}_i \in \mathbb{R}^d$ is the vector of features of the i^{th} sample, $y_i = 1, ..., C$ 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

The hypothesis of mixture models is that a given sample is the realization of a random vector which distribution is a mixture (convex combination) of several class conditioned distribution:

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

where π_c is the prior i.e. the proportion of class c and f_c a probability density function parametrized by θ.

The Gaussian mixture model (GMM) assumes that each f_c is , conditionally to c, a Gaussian distribution of parameters $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ and so $f_c(\mathbf{x}|\theta)$ can be written as

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

In supervised learning the class parameters $\boldsymbol{\mu}_c$, $\boldsymbol{\Sigma}_c$ and π_c are usually estimated through the conventional unbiased empirical estimators:

$$\hat{\pi}_c = \frac{n_c}{n}, \tag{2}$$

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{n_c}\sum_{\{i|y_i=c\}} \mathbf{x}_i, \tag{3}$$

$$\hat{\boldsymbol{\Sigma}}_c = \frac{1}{(n_c-1)}\sum_{\{i|y_i=c\}} (\mathbf{x}_i-\boldsymbol{\mu}_c)(x_i-\boldsymbol{\mu}_c)^t. \tag{4}$$

To predict the class of a new sample, the maximum a posteriori (MAP) rule is used and thus, using Bayes' law, the decision rule is

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

Identifying $p(c)$ as π_c and $p(x|c)$ as $f_c(x|\theta)$ and by taking the log, a simplified decision rule is obtained

$$\begin{aligned} Q_c(\mathbf{x}) &= 2\log\left(p(c)p(\mathbf{x}|c)\right) \\ &= -(\mathbf{x}-\boldsymbol{\mu}_c)^t\boldsymbol{\Sigma}_c^{-1}(\mathbf{x}-\boldsymbol{\mu}_c) \\ &\quad - \log(|\boldsymbol{\Sigma}_c|) + 2\log(\pi_c) - d\log(2\pi). \end{aligned} \tag{5}$$

It is important to notice that the inverse and the determinant of the covariance matrix are key elements of the decision function. The estimations of these elements suffer from the curse of dimensionality [4]. More precisely, the number of parameters to estimate increases quadratically relatively to the number of features as illustrated by Figure 1. The minimum number of

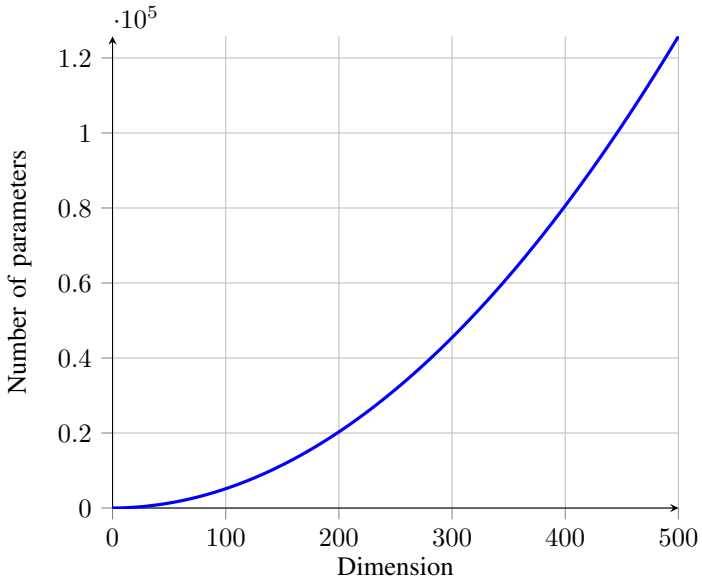


Fig. 1. Number of parameters by class in function of dimension.

samples requires to estimate the model is equal to the number of parameters and it can be an issue in some case. For example, hyperspectral data is a case of high dimensional samples and, in general, very few labeled samples are available because of the difficulty and the cost to collect ground-truth.

A lack of samples induces badly-conditioned covariance matrices and so unstable inversion and computation of the determinant. There are two major solutions to this problem. First option is to use a regularization method to stabilize the inversion of the covariance matrices [15]. Second option is to use a features extraction/selection method in order to reduce the dimensionality of the samples. In this study, the latter option is explored and a feature selection method named sequential forward features selection is presented in the next Section.

B. Features selection

The objective of feature selection is to retrieve the most compact representation of the data with minimum loss of information. In deleting the redundant information, feature selection allows to reduce dimension and thus avoid statistical issue, increase algorithm speed and limit storage requirements. Moreover, it adds several beneficial side effects. It improves data understanding by identify where is the most relevant information and it is then possible to save resources when organizing a new data acquisition.

It is important to underline that the proposed method is a *selection* method and not an *extraction* method. It means the subset of variables obtained is composed of actual variables of the original set and not variables built as combination of several others as it is the case for example with PCA. This choice is made to assure an easier interpretation by the user of the obtained subset of variables.

Features selection algorithms may be divided into three types. The first type called filter method is based uniquely on data analysis. Features are ranked according to a statistical analysis of the data. For example, the Principal Component Analysis (PCA) described in [7] is a typical filter method and there is numerous other methods [16], [17], [18].

The second sort are known as wrapper methods which can be seen as search method to determine the best subset of variables for a given learning model. As exhaustive searches are out of question in a practical amount of time, numerous search strategies have been designed some optimal under particular hypothesis [11] and other suboptimal but easier to set up [12], [13]. The advantage of such methods compare to filter methods is that they are dedicated to a particular model but on the other hand, as they require the training of multiple models to test various set of variables, they tend to be slower.

The third type corresponds to the embedded method which do not separate the features selection process from the learning algorithm and allow interaction between the two processes. The basic example of such method is the decision tree algorithm

in which a feature is selected for the creation of each node. Embedded methods also exist for other models, e.g. SVM [19] [20]. The selection method proposed in this work is a wrapper method associated to GMM models. Thus, in order to set a wrapper method, two elements are needed, a function to rank the various features which is defined in Section II-B1 and a search strategy. Two search algorithms are presented: the sequential forward features selection method (Section II-B2a) and the sequential floating forward feature selection method (II-B2b) the second being a more complex variation of the first.

1) Criterion function:

Criterion functions aim to evaluate either a rate of correct classification or the separability/similarity of class distributions. These functions are used to estimate which sets of variables are the best to represent data, to assure the separability of the classes and to perform classification. The choice of a criterion function is the choice of a way to compare sets of variables.

a) Measures of correct classification:

As described in [21] (chapter 4), all this measures of good classification are based on an error matrix M called confusion matrix which is defined so that M_{ij} is the number of samples of class i classified as class j. The confusion matrix allows the computation of several interesting values relatively to each class:

- the number of True Positive (TP) corresponding to good prediction;
- the number of True Negative (TN) corresponding to good classification of the other class;
- the number of False Negative (FN) corresponding to the samples of the class labeled as an other class;
- the number of False Positive (FP) corresponding to the samples wrongly classified as part of of this class.

Figure 2 illustrates the definition of this values.

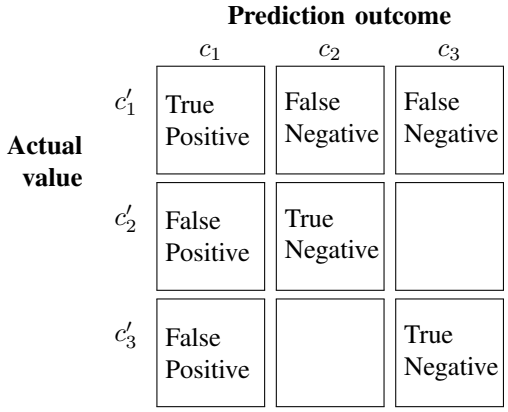


Fig. 2. Confusion matrix with TP, TN, FP and FN relatively to c_1

The overall accuracy (OA) is the rate of the number of samples with the correct predicted label over the total number of samples. 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.

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).

$$OA = \frac{\text{TP}}{n}. \tag{6}$$

$$K = \frac{pa - pr}{1 - pr}. \tag{7}$$

$$\text{F1mean} = \frac{2\text{TP}}{2\text{TP} + \text{FN} + \text{FP}}. \tag{8}$$

where TP stands for True Positive, FN for False Negative, FP for False Positive, pa is the probability of agreement defined by $pa = OA$ and pr the probability of random agreement defined by $pr = \sum_{c=1}^{C} \frac{TP_c}{FP_c} \frac{TP_c}{FN_c}$.

In order to estimate classification rate, a cross-validation process over training set is used [22]. The training set is divided in k folds, then, the training is done with $(k-1)$ folds and the performances are estimated with the remaining fold.

b) Similarity between distributions:

The second type of criterion functions is a measure of similarity between two distributions. These measures are called divergence function and are defined so that, if S is a space of all probability distribution with same support, it satisfies

$$\forall (p,q) \in S, \mathrm{Div}(p,q) \geq 0,$$
$$\mathrm{Div}(p,q) = 0 \Leftrightarrow p = q.$$

More specifically, focus is made on two particular divergences: the KullbackLeibler divergence and the JeffriesMatusita 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 calculus 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 [23]. The formal definition is

$$\mathrm{Div}_{KL}(c_i, c_j) = \int_{\mathbf{x}} p(\mathbf{x}|c_i) \ln \left(\frac{p(\mathbf{x}|c_i)}{p(\mathbf{x}|c_j)} \right) d\mathbf{x}. \tag{9}$$

And in the case of Gaussian model, it can be rewritten as follows

$$\begin{aligned} \mathrm{Div}_{KL}(c_i, c_j) = \frac{1}{2} \Bigg(& \mathrm{Tr}(\mathbf{\Sigma}_{c_i}^{-1} \mathbf{\Sigma}_{c_j}) \\ & + (\boldsymbol{\mu}_{c_i} - \boldsymbol{\mu}_{c_j})^t \mathbf{\Sigma}_{c_i}^{-1} (\boldsymbol{\mu}_{c_i} - \boldsymbol{\mu}_{c_j}) - d + \log \left(\frac{|\mathbf{\Sigma}_{c_i}|}{|\mathbf{\Sigma}_{c_j}|} \right) \Bigg), \end{aligned} \tag{10}$$

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

It can be noticed that the KL divergence is not symmetric, i.e. $\mathrm{Div}_{KL}(c_i, c_j) \neq \mathrm{Div}_{KL}(c_j, c_i)$, and so the symmetrized version is used to compute the criterion function. In the case of Gaussian model, the symmetrization induces the following simplification of the formula

$$\begin{aligned} \mathrm{SKL}_{ij} &= \mathrm{Div}_{KL}(c_i, c_j) + \mathrm{Div}_{KL}(c_j, c_i) \\ &= \frac{1}{2} \Bigg(\mathrm{Tr}(\mathbf{\Sigma}_{c_i}^{-1} \mathbf{\Sigma}_{c_j} + \mathbf{\Sigma}_{c_j}^{-1} \mathbf{\Sigma}_{c_i}) \\ &\quad + (\boldsymbol{\mu}_{c_i} - \boldsymbol{\mu}_{c_j})^t (\mathbf{\Sigma}_{c_i}^{-1} + \mathbf{\Sigma}_{c_j}^{-1}) (\boldsymbol{\mu}_{c_i} - \boldsymbol{\mu}_{c_j}) - 2d \Bigg). \end{aligned} \tag{11}$$

Moreover, the divergence is computed between two classes and to obtain a unique value the weighted mean of divergence measures is taken as proposed in [16]:

$$\mathrm{C}_{SKL} = \sum_{i=1}^{C} \sum_{j=i+1}^{C} \pi_{c_i} \pi_{c_j} \mathrm{SKL}_{ij}. \tag{12}$$

The Bhattacharyya distance is defined as follows

$$\mathrm{B}_{ij} = -\ln \left(\int_{\mathbf{x}} \sqrt{p(\mathbf{x}|c_i) p(\mathbf{x}|c_j)} d\mathbf{x} \right). \tag{13}$$

And in the case of Gaussian model:

$$\begin{aligned} \mathrm{B}_{ij} = & \frac{1}{8} (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^t \left(\frac{\mathbf{\Sigma}_i + \mathbf{\Sigma}_j}{2} \right)^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) \\ & + \frac{1}{2} \ln \left(\frac{|\mathbf{\Sigma}_i + \mathbf{\Sigma}_j|}{\sqrt{|\mathbf{\Sigma}_i||\mathbf{\Sigma}_j|}} \right). \end{aligned} \tag{14}$$

The JeffriesMatusita distance is a measure based on the Bhattacharyya distance. It saturates if the separability between the

TABLE I
SUMMARY OF THE VARIOUS CRITERION FUNCTIONS.

Criterion	Divergence	Classification rate	Gaussian hypothesis	Speed
Overall accuracy		✓	0	-
Cohen's kappa		✓	0	-
F1 mean		✓	0	-
Kullback-Leibler divergence	✓		++	+
Jeffries-Matusita distance	✓		++	+

two distribution increases [24]. The JM distance is defined according to

$$\mathrm{JM}_{ij} = \sqrt{\int_{\mathbf{x}} \left[\sqrt{p(\mathbf{x}|c_i)} - \sqrt{p(\mathbf{x}|c_j)}\right]^2 d\mathbf{x}}. \tag{15}$$

And the JeffriesMatusita distance can be rewritten according to the Bhattacharyya distance

$$\mathrm{JM}_{ij} = \sqrt{2\{1 - \exp[-B_{ij}]\}}. \tag{16}$$

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

$$\mathrm{C}_{JM} = \sum_{i=1}^{C} \sum_{j=i+1}^{C} \pi_{c_i} \pi_{c_j} \mathrm{JM}_{ij}. \tag{17}$$

According to [24], it is interesting to notice that the KL divergence increases quadratically with respect to the distance between the mean vectors of the class distributions whereas the measures of correct classification asymptotically tends to one when distributions are perfectly separable. On the contrary, the JM distance tends to saturate as the measures of correct classification. Table I summarized the presented criterion functions and their characteristics.

2) Selection method:

a) Sequential forward features selection: The Sequential Forward Selection (SFS) starts with an empty set of selected features. Then, it computes at each step for all the remaining features the value of the criterion function J chosen among the ones presented in Table I when the feature is added to the pool of selected features. The feature that maximizes the criterion function is definitively added to the pool of selected features and it moves to the next iteration. The algorithm stops when a given number of variables *maxVarNb* has been selected. The Algorithm 1 presents the process in details.

The advantage of this search algorithm is its reasonable processing time. The trade-off is that the result is a suboptimal solution in the sense that an untested subset of variable could lead to better classification result.

Algorithm 1 Sequential forward features selection

Require: Ω, J, maxVarNb
$\Omega = \emptyset$
$F = \{\text{all variables}\}$
while card$(\Omega) \leq maxVarNb$ **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

b) Sequential floating forward feature selection: The Sequential Floating Forward Selection (SFFS) [13] is actually 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. At the end of the SBS step, the value of the criterion function is compared to the best value ever obtained with a set of features of the same size and if the new value is better the feature put into question is effectively taken away and the next step is again a SBS but if the new value is not better the SBS step is forgotten and it moves to the next SFS step. The algorithm stops when a given number of features *maxVarNb* has been selected. The Algorithm 2 sums up the method.

This SFFS algorithm eventually tests 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
$\Omega = \overbrace{(\emptyset, ..., \emptyset)}^{maxVarNb}$
$F = \{\text{all variables}\}$
$k = 0$
while $k \neq$ maxVarNb **do**
  for all $f_i \in F$ **do**
    $R_i = J(\{\Omega_k + f_i\})$
  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 $f_i \in \Omega_k$ **do**
        $R_i = J(\{\Omega_k \setminus f_i\})$
      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}

III. Efficient implementation

A. Statistical update rules

In the presented method, the most costly operation is the evaluation of the criterion function which needs to be done for each possible subset of variables augmented by one of the candidate variable. To compute these values, the most costly operation are the inversion of the covariance matrix and the computation of its determinant. Both of these operations have a $O(n^3)$ computational complexity. Additionaly, the estimation of classification rate with cross-validation is a heavy process which take a consequent amount of time.

A major contribution of this work is the optimization in term of computational efficiency. More precisely, three steps of the proposed method has been upgraded in order to reduce computational time. First of all, the GMM model is learned only once using samples. When a covariance matrix or a mean vector of a reduced set of variables is required, it is obtained from the global model learned at the beginning by marginalization.

Secondly, when a cross-validation is performed, submodels, i.e. GMM model trained with $(n_{cv} - 1)$ folds, are not learned using the $(n_{cv} - 1)$ folds but, instead, are derived from the global model and from the covariance matrices and mean vectors of the fold used for validation for the given submodel. The process is described in details in Section III-A1.

Finally and most importantly, when variables are tested one by one during a selection step, costly operations are made when computing criterion functions for each variable especially the computation of the inverse of covariance matrices and its

determinant. Several update rules are set up which allow to compute this inverse and determinant only once to test all variables. These update rules are presented in Section III-A2.

1) Update for cross validation: Based on [10], 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 model 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 submodels.

The following formulae can be obtained

Proposition 1. *(Cross-validation mean update)*

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

Proposition 2. *(Cross-validation covariance matrix update)*

$$\begin{aligned}\boldsymbol{\Sigma}_c^{n_c-\nu_c} =& \frac{1}{n_c-\nu_c-1}\Bigg((n_c-1)\boldsymbol{\Sigma}_c^{n_c} - (\nu_c-1)\boldsymbol{\Sigma}_c^{\nu_c} \\ &- \frac{n_c\nu_c}{(n_c-\nu_c)}(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})^t\Bigg)\end{aligned}$$

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, exposants on $\boldsymbol{\Sigma}_c$ and $\boldsymbol{\nu}_c$ denotes the set of samples used to compute them.

2) Criterion function computation: At each iteration the SFS and SFFS algorithms compute the value of a criterion function for every possible set of features composed of the selected ones augmented by one of the remaining features. One of the main achievements of this work is to reduce the computational time in factorizing the computation of the inverse and determinant of the covariance matrix..

In the remaining of the paper, $\boldsymbol{\Sigma}^{(k-1)}$ is the covariance matrix of the $(k-1)^{th}$ iteration, i.e., the covariance matrix of the selected features and $\boldsymbol{\Sigma}^{(k)}$ is a covariance matrix at the k^{th} iteration, i.e., the covariance matrix of an augmented set. Then, the inverse of the covariance matrix $(\boldsymbol{\Sigma}^{(k)})^{-1}$, the quadratical term $(\mathbf{x}^{(k)})^t(\boldsymbol{\Sigma}^{(k)})^{-1}\mathbf{x}^{(k)}$ and the determinant $\log|\boldsymbol{\Sigma}^{(k)}|$ can be expressed in function of terms of the $(k-1)^{th}$ iteration. These calculi use the fact that the covariance matrix is a positive definite symmetric matrix to simplify formulae using block matrices [25] (chapter 9.2). These update rules are summed up hereafter.

As $\boldsymbol{\Sigma}^{(k)}$ is a positive definite symmetric matrix, the following notation can be used

$$\boldsymbol{\Sigma}^{(k)} = \begin{bmatrix} \boldsymbol{\Sigma}^{(k-1)} & \mathbf{u} \\ \mathbf{u}^t & \sigma_{kk} \end{bmatrix},$$

where $\mathbf{u}$ is the k^{th} column of the matrix without the diagonal element i.e. $\mathbf{u}_i = \boldsymbol{\Sigma}_{i,k}^{(k)}$ with $i \in [1, k-1]$.

Using the formula of the inverse of a block matrix, the following formula expressing $(\boldsymbol{\Sigma}^{(k)})^{-1}$ in function of $(\boldsymbol{\Sigma}^{(k-1)})^{-1}$ is obtained

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

$$(\boldsymbol{\Sigma}^{(k)})^{-1} = \begin{bmatrix} A & v \\ v^t & \frac{1}{\alpha} \end{bmatrix}$$

where $A = (\boldsymbol{\Sigma}^{(k-1)})^{-1} + \frac{1}{\alpha}(\boldsymbol{\Sigma}^{(k-1)})^{-1}\mathbf{u}\mathbf{u}^t(\boldsymbol{\Sigma}^{(k-1)})^{-1}$, $v = -\frac{1}{\alpha}(\boldsymbol{\Sigma}^{(k-1)})^{-1}\mathbf{u}$ and $\alpha = \sigma_{kk} - \mathbf{u}^t(\boldsymbol{\Sigma}^{(k-1)})^{-1}\mathbf{u}$. Then, $(\boldsymbol{\Sigma}^{(k-1)})^{-1}$ is computed only once and this update update formulae is used to obtain all the $(\boldsymbol{\Sigma}^{(k)})^{-1}$ corresponding to all the possible augmented set of a given selection iteration.

In the case of a backward step in SFFS algorithm, the formula is inverted in order to compute $(\boldsymbol{\Sigma}^{(k-1)})^{-1}$ knowing $(\boldsymbol{\Sigma}^{(k)})^{-1}$. The update rule becomes (proof in Appendix A)

Proposition 4. *(Backward update rule for inverse of covariance matrix)*

$$(\mathbf{\Sigma}^{(k-1)})^{-1} = \underbrace{\mathbf{A}}_{\substack{\textit{computed once}\\ \textit{per selection step}}} - \underbrace{\alpha \mathbf{v}\mathbf{v}^t}_{\substack{\textit{computed for}\\ \textit{each augmented set}}}$$

A formula can also be deduced for the quadratical term of the decision function. Noting $(\mathbf{x}^{(k)})^t = \begin{bmatrix} (\mathbf{x}^{(k-1)})^t & x^k \end{bmatrix}$, the following proposition is obtained (proof in Appendix A)

Proposition 5. *(Update rule for quadratical term)*

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

Finally, using the formula of the determinant of a block matrix and after taking the log, a last formula is produced

Proposition 6. *(Update rule for logdet)*

$$\log\left(|\mathbf{\Sigma}^{(k)}|\right) = \underbrace{\log\left(|\mathbf{\Sigma}^{(k-1)}|\right)}_{\substack{\textit{computed once}\\ \textit{per selection step}}} + \underbrace{\log\alpha}_{\substack{\textit{computed for}\\ \textit{each augmented set}}}$$

Now using all update rules, criterion functions can be rewritten as follows

$$\begin{aligned} Q(\mathbf{x}) = &- \underbrace{(\mathbf{x}^{(k-1)} - \boldsymbol{\mu}^{(k-1)})^t(\mathbf{\Sigma}^{(k-1)})^{-1}(\mathbf{x}^{(k-1)} - \boldsymbol{\mu}^{(k-1)})}_{\substack{\text{computed once}\\ \text{per selection step}}} \\ &- \underbrace{\log\left(|\mathbf{\Sigma}^{(k-1)}|\right) + 2\log(\pi_c) + k\log(2\pi)}_{\substack{\text{computed once}\\ \text{per selection step}}} \\ &- \underbrace{\alpha\left(\begin{bmatrix} \mathbf{v}^t & \frac{1}{\alpha} \end{bmatrix}(\mathbf{x}^{(k)} - \boldsymbol{\mu}^{(k)})\right)^2 - \log\alpha}_{\substack{\text{computed for}\\ \text{each augmented set}}}; \end{aligned} \tag{18}$$

$$\begin{aligned} \mathrm{SKL}_{ij} = \frac{1}{2}\Bigg(&\underbrace{\mathrm{Tr}\left((\mathbf{\Sigma}_{c_i}^{(k)})^{-1}\mathbf{\Sigma}_{c_j}^{(k)} + (\mathbf{\Sigma}_{c_j}^{(k)})^{-1}\mathbf{\Sigma}_{c_i}^{(k)}\right)}_{\substack{\text{computed for}\\ \text{each augmented set}}} \\ &+ \underbrace{\alpha(\begin{bmatrix} \mathbf{v_i}^t & \frac{1}{\alpha_i} \end{bmatrix}(\boldsymbol{\mu}_{c_i}^{(k)} - \boldsymbol{\mu}_{c_j}^{(k)}))^2}_{\substack{\text{computed for}\\ \text{each augmented set}}} \\ &+ \underbrace{\alpha(\begin{bmatrix} \mathbf{v_j}^t & \frac{1}{\alpha_j} \end{bmatrix}(\boldsymbol{\mu}_{c_i}^{(k)} - \boldsymbol{\mu}_{c_j}^{(k)}))^2 - 2k}_{\substack{\text{computed for}\\ \text{each augmented set}}} \\ &+ \underbrace{(\boldsymbol{\mu}_{c_i}^{(k-1)} - \boldsymbol{\mu}_{c_j}^{(k-1)})^t((\mathbf{\Sigma}_{c_i}^{(k-1)})^{-1} + (\mathbf{\Sigma}_{c_j}^{(k-1)})^{-1})(\boldsymbol{\mu}_{c_i}^{(k-1)} - \boldsymbol{\mu}_{c_j}^{(k-1)})}_{\substack{\text{computed once}\\ \text{per selection step}}}\Bigg), \end{aligned} \tag{19}$$

with $(\mathbf{\Sigma}_{c_i}^{(k)})^{-1}$ computed with Proposition 3;

$$
\begin{aligned}
\mathrm{B}_{ij} = & \underbrace{\frac{1}{4}(\boldsymbol{\mu}_i^{(k-1)} - \boldsymbol{\mu}_j^{(k-1)})^t (\tilde{\boldsymbol{\Sigma}}^{(k-1)})^{-1} (\boldsymbol{\mu}_i^{(k-1)} - \boldsymbol{\mu}_j^{(k-1)})}_{\substack{\text{computed once} \\ \text{per selection step}}} \\
& + \underbrace{\frac{1}{2} \ln \left(\frac{|\tilde{\boldsymbol{\Sigma}}^{(k-1)}|}{\sqrt{|\boldsymbol{\Sigma}_i^{(k-1)}||\boldsymbol{\Sigma}_j^{(k-1)}|}} \right)}_{\substack{\text{computed once} \\ \text{per selection step}}} \\
& + \underbrace{\frac{1}{4}\tilde{\alpha}(\begin{bmatrix} \tilde{\mathbf{v}}^t & \frac{1}{\tilde{\alpha}} \end{bmatrix} (\boldsymbol{\mu}_i^{(k)} - \boldsymbol{\mu}_j^{(k)}))^2 + \frac{1}{2} \ln \left(\frac{\tilde{\alpha}}{\sqrt{\alpha_i \alpha_j}} \right)}_{\substack{\text{computed for} \\ \text{each augmented set}}},
\end{aligned} \tag{20}
$$

where $\tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j$.

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

B. Numerical issues

During an iteration of the selection algorithm, it still needs to invert the covariance matrix of the previous iteration. In order to so, a decomposition in eigenvalues and eigenvectors is performed using algorithm specific for symmetric matrices. More precisely, In other words, $\boldsymbol{\Sigma}$ is decomposed s.t. $\boldsymbol{\Sigma} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^t$ with $\boldsymbol{\Lambda}$ the diagonal matrix of eigenvalues and $\mathbf{Q}$ the matrix of eigenvectors.

This decomposition has several advantages. First, it is very simple to compute the determinant once the eigenvalues are available because the determinant is equal to the product of these values. Secondly, it is possible to check the eigenvalues and so to assure a better computational stability. Due to the curse of dimensionality, the covariance matrix is sometimes badly conditioned and the result is that some eigenvalues are really small (below computational precision). The decomposition helps to check their actual values and if they are below EPS_FLT which is the machine maximum precision for a float, these eigenvalues are set to EPS_FLT. For the same reason, the constant α used in update rules is also thresholded to EPS_FLT. Algorithm 3 details when computational stability is checked.

C. OTB external module

The Orfeo Toolbox is an open-source library for remote sensing image processing. This library allows developers to code external modules and make them available to the community. It is to note that the toolbox already implements several classifiers inherited mostly from OpenCV. Nevertheless, the GMM classifier currently in place is not robust to high dimensionality and that is why a new GMM model has been developed.

The developed module is available on Github [1] and is a fork of the template for remote module furnished by OTB developers.

The GMM classifier class called *GMMMachineLearningModel* inherits from the OTB class *MachineLearningModel* which is a basic class used for all classifier in the OTB. The *MachineLearningModel* class enables the management of a list of samples used for training and also declares the virtual methods inherited: *Train()*, *Predict()*, *Save()*, *Load()*.

The *GMMMachineLearningModel* class implements all these virtual methods. The *Train()* aims to train the classifier using a list of samples set as a class member. Then, it is possible to use the method *Predict()* to classify a sample and optional get the confidence in the prediction. The methods *Save()* and *Load()* obviously are used to save and load a trained model.

In addition to these inherited methods, two noticeable methods are implemented. The *Decomposition()* method perform a decomposition of a symmetric matrix in eigenvalues and eigenvectors using a function of the VNL library. It is to notice that this method check the value of the extracted eigenvalues and minor them to EPSILON_FLT (or EPSILON_DBL) which corresponds to computational precision. The idea is to limit computational instability when parameters are badly-estimated

[1] https://github.com/Laadr/otbExternalFastFeaturesSelection

Algorithm 3 Sequential floating forward features selection with updates

Require: J, maxVarNb
$\Omega = \overbrace{(\emptyset, ..., \emptyset)}^{maxVarNb}$
$F = \{\text{all variables}\}$
$k = 0$
while $k \neq$ maxVarNb **do**
  Diagonalize $\mathbf{\Sigma}^{(k-1)} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^t$
  for all $\Lambda_{i,i}$ **do** $\Lambda_{i,i} = \max(\text{EPS\_FLT}, \Lambda_{i,i})$
  Precompute $(\mathbf{\Sigma}^{(k-1)})^{-1}$, $(\mathbf{x}^{(k-1)} - \boldsymbol{\mu}^{(k-1)})^t(\mathbf{\Sigma}^{(k-1)})^{-1}(\mathbf{x}^{(k-1)} - \boldsymbol{\mu}^{(k-1)})$ and $\log\left(|\mathbf{\Sigma}^{(k-1)}|\right)$ using Propositions 3, 5 and 6
  for all $f_i \in F$ **do**
    Compute update constant α
    $\alpha = \max(\text{EPS\_FLT}, \alpha)$
    $R_i = J(\{\Omega_k + f_i\})$ using Equations 18, 19 or 20
  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**
      Diagonalize $\mathbf{\Sigma}^{(k-1)} == \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^t$
      for all $\Lambda_{i,i}$ **do** $\Lambda_{i,i} = \max(\text{EPS\_FLT}, \Lambda_{i,i})$
      Compute $(\mathbf{\Sigma}^{(k-1)})^{-1}$, $(\mathbf{x}^{(k-1)} - \boldsymbol{\mu}^{(k-1)})^t(\mathbf{\Sigma}^{(k-1)})^{-1}(\mathbf{x}^{(k-1)} - \boldsymbol{\mu}^{(k-1)})$ and $\log\left(|\mathbf{\Sigma}^{(k-1)}|\right)$ using Propositions 3, 5 and 6
      for all $f_i \in \Omega_k$ **do**
        Compute update constant α
        $\alpha = \max(\text{EPS\_FLT}, \alpha)$
        $R_i = J(\{\Omega_k \setminus f_i\})$ using Equations 18, 19 or 20
      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}

due to an insufficient amount of training samples. The second interesting method is *TrainTau()* which concerns an optional ridge regularization. Given a set of possible regularization constant τ, this method selects the most efficient value in estimating classification rate with cross-validation.

To implement the selection algorithm, a new class *GMMSelectionMachineLearningModel* inheriting from *GMMMachine-LearningModel* is defined. This class keeps the same six methods inherited from *MachineLearningModel* but reimplements *Predict()* to predict with a reduced set of features and *Save()*/*Load()* to use a second file to handle the results of the selection.

The other methods available in class *GMMSelectionMachineLearningModel* are used to perform the selection algorithms presented in Section II. The method *Selection()* aims to set the various parameter of the selection and then calls either the *ForwardSelection()* method to use forward feature selection or the *FloatingForwardSelection()* method to use SFFS. Then, 3 different methods are used to evaluate criterion functions during selection. Given a pool of variables available for selection, these functions evaluate the criterion functions for all the possible augmented set. The *ComputeJM()* method is used for Jeffries-Matusita distance, *ComputeDivKL()* for KullbackLeibler divergence and *ComputeClassifRate()* for the three correct classification criteria. It is to notice that the correct classification criteria are evaluated using cross-validation.

In order to provide an easy way to use the developed algorithm, OTB applications has been developed. These applications allow the user to use and parametrize the algorithm from command line. Three applications are built.

The application *otbcli_TrainGMMApp* creates and trains a model from class *GMMMachineLearningModel* and the application *otbcli_TrainGMMSelectionApp* does the same for a model of class *GMMSelectionMachineLearningModel*. These two applications use a raster image and shapefile with groundtruth as input and create a model file.

The last application *otbcli_PredictGMMApp* takes a raster image as input and a model file and perform classification of the image. It generates a classification map in an image file and optionally a confidence map. An other application is available in

the OTB to compute classification rate from the image and a groundtruth file.

IV. Experimental results

A. Method

The aim of the experiments is to compare the proposed method to some standard classifiers. In order to do it, OTB commandline application are used. The following classifier are tested:

- a k-nearest-neighbors classifier (KNN) with OTB default parameters (32 number of neighbors);
- a Random Forest classifier with parameters optimized by gridsearch (200 trees, 40 max depth, 50 size of subset of variables);
- a GMM classifier with ridge regularization (GMM ridge) with regularization constant optimized by gridsearch.

The GMM classifier is part of the external module described in Section III-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).

The results obtained with two different datasets are presented. The first dataset is an airborne hyperspectral dataset and second is a remote sensing dataset made of heterogeneous spatial features. Training set has been created with an equal number of samples for each class and additionally a spatial stratification has been performed. Several size of training set has been tested. For 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.

For SFS and SFFS selection, the number of variables to select is set to 30 for the Aisa datset and 60 for the Potsdam dataset and then the number of variables which maximizes the criterion function is used to perform classification.

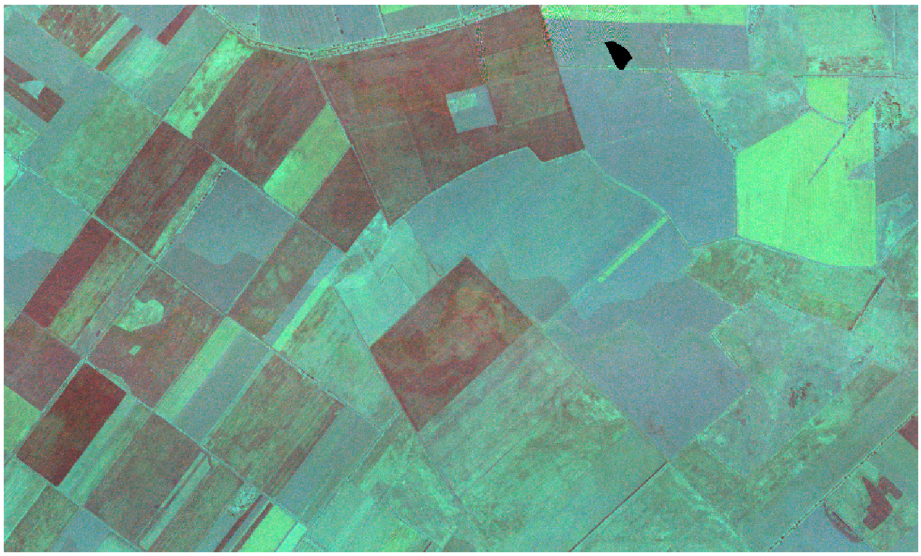
The classification rate is presented using Cohen's kappa but it has to be noticed the results were similar when looking at the mean F1-score. Processing time has been evaluated on a desktop computer with 4Gb of RAM and Intel(R) Core(TM) i5-3570 CPU 3.40GHz x 4 processor. All results has been estimated with multiple repetitions (20 trials with Aisa dataset, 5 trials with Potsdam dataset).

B. Aisa dataset

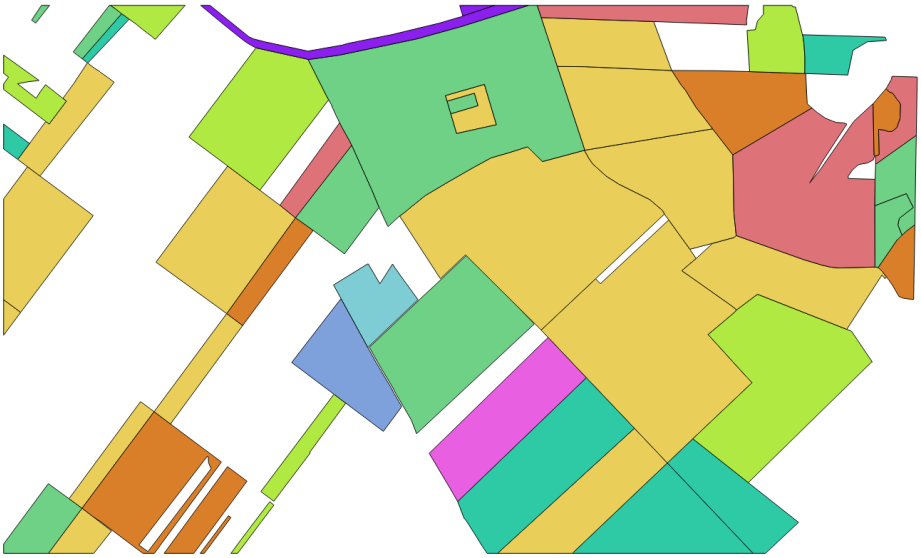
The Aisa dataset has been acquired by the AISA Eagle sensor during a flight campaign over Heves, Hungary. It contains 252 bands ranging from 395 to 975 nm. 16 classes have been defined for a total of 361,971 referenced pixels. Figure 3 shows a colored composition of the multispectral image and the shapefile representing groundtruth and Table II presents the repartition of the various classes.

When creating training and validation sets, special care is taken to assure that training samples are picked in distinct areas than test samples. In order to do it, 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. Moreover 20 trials are run each time with a different training set (different polygons). An example of training and validation set is shown in Figure 4. Additionally, it is important to understand that only a subpart of this training set is actually used to train the model, a given number of samples for each class. Table III presents the results of the experiment with mean and standard deviation over the 20 trials and Table IV the corresponding processing time.

The results show that, on this dataset, GMM classifiers with selection get the best classification rate. Especially with very few training samples, GMM classifiers with selection seems to outperform the others. In term of computational time, the GMM classifiers are as expected very fast for classification and also for training if the criterion function is not a classification rate. The computation of the kappa seems to suffer from a lack of parallelization which could be improved. 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. In this case, using J-M distance as criterion and SFS as search strategy is the best choice in term of time efficiency.



(a)



(b)

Fig. 3. Aisa dataset: **(a)** colored composition of the image, **(b)** groundtruth.

Nevertheless, good performance are also obtained with Random Forest which becomes more competitive when the number of samples increases. The KNN classifier and the GMM classifier with ridge regularization are totally outperform both in term of classification rate and of processing time.

C. 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 is composed of aerial images of the urban area of Potsdam. The area is cut into 38 patches of 6000x6000 pixels with a resolution of 5cm by pixel and 4 channels are available: Red, Blue, Green and

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

TABLE II
REPARTITION OF CLASSES IN 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

TABLE III
RESULTS OF CLASSIFICATION WITH SAMPLING ON SEPARATE POLYGONS AND 20 TRIALS (STANDARD DEVIATION IN PARENTHESIS).

	Cohen's kappa		
# samples by class	250	500	1000
GMM SFS kappa	0.684 (0.025)	0.698 (0.027)	0.711 (0.028)
GMM SFS JM	0.680 (0.028)	0.700 (0.028)	0.710 (0.030)
GMM SFFS JM	0.680 (0.028)	0.700 (0.028)	0.710 (0.030)
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)

TABLE IV
MEAN PROCESSING TIME FOR TRAINING AND CLASSIFICATION FOR RESULTS OF TABLE III.

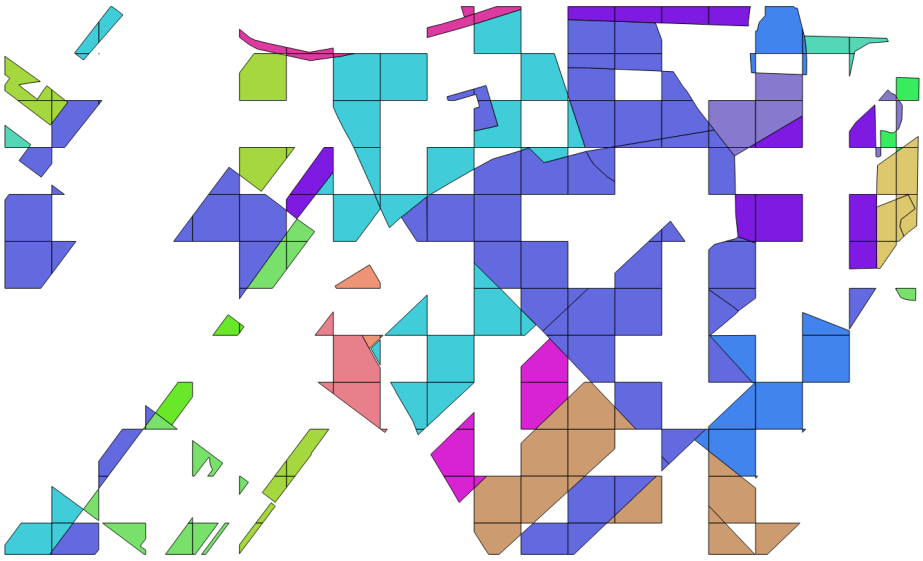
	Training time (s)			**Classification time (s)**		
# samples by class	250	500	1000	250	500	1000
GMM SFS kappa	257	496	955	7.7	8.6	8.7
GMM SFS JM	8.6	8.9	9.1	9.7	9.8	9.6
GMM SFFS JM	8.8	9.0	9.3	9.7	9.8	9.8
GMM ridge	71.7	105	167	530	530	530
KNN	8.9	19.6	59.7	387	639	887
Random Forest	24.5	49.3	105	33.0	41.7	45.9

Infrared (RGBIR). A Digital Surface Model with same resolution is also provided and a so-called normalized DSM representing the height above ground. The groundtruth for 24 tiles are provided with 6 classes: Low vegetation, High vegetation, Impervious surfaces, Buildings, Cars, Clutter. Table V summarizes the number of samples of each class in the two images used during experimentations. Figure 5 shows the complete area and Figure 6 a particular patch.

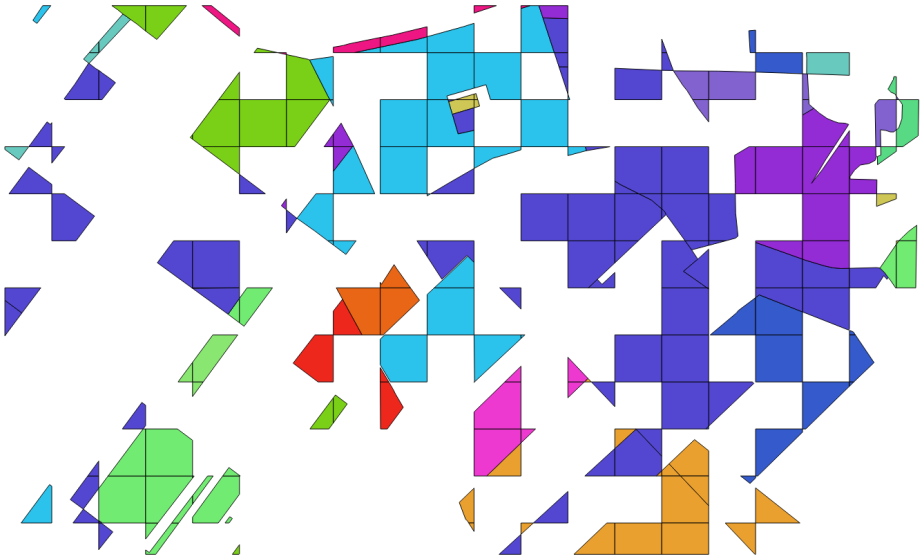
In the experiment, the following features are computed using the RGBIR images similar to [26]:

- Radiometric indexes: NDVI, TNDVI, RVI, SAVI, TSAVI, MSAVI, MSAVI2, GEMI, IPVI, NDWI2, NDTI, RI, CI, BI, BI2 (15 features) [3];
- Morphological profile with reconstruction of each band with a disk of radius 5, 9, 13, 17, 21, 25, 29, 33, 37 and 41 (80 features) [26];

[3]https://www.orfeo-toolbox.org//Applications/RadiometricIndices.html



(a)



(b)

Fig. 4. Aisa dataset: **(a)** training polygons of first trial, **(b)** test polygons of first trial.

TABLE V
REPARTITION OF CLASSES IN AISA DATASET.

Class	Number of samples in 5_11	Number of samples in 5_12
Clutter	1,078,611	812,038
Trees	4,493,295	2,132,368
Cars	900,076	1,101,541
Buildings	13,469,575	17,501,421
Low vegetation	4,718,219	3,210,596
Impervious surfaces	11,340,224	11,242,036

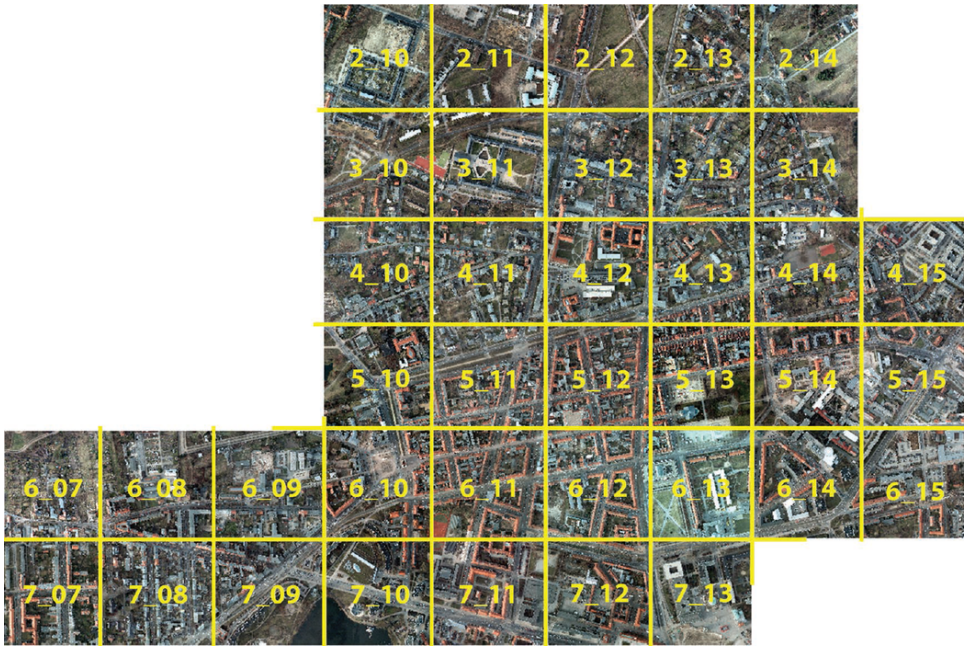


Fig. 5. Full Potsdam dataset.

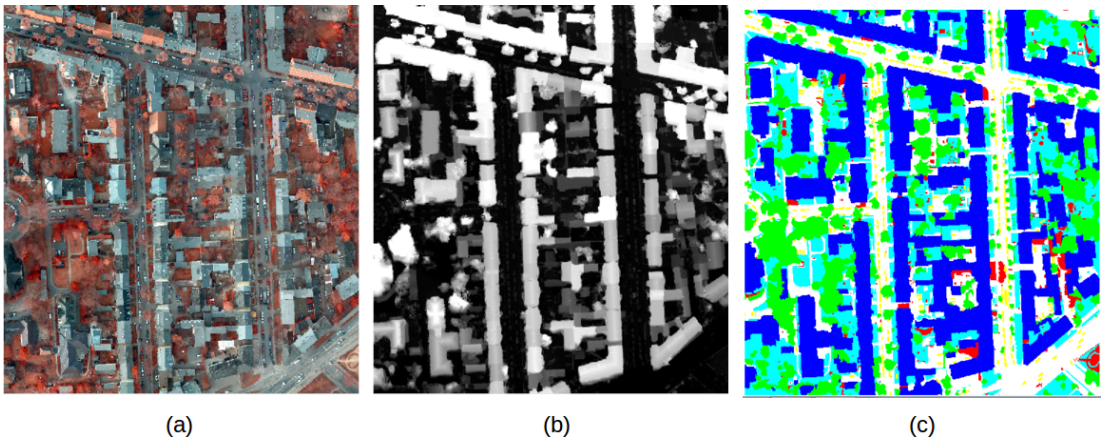


Fig. 6. Example of patch with **(a)** orthophoto, **(b)** DSM and **(c)** groundtruth.

- Attribute profile of each band with area as attribute and 1000, 2000, 5000, 10000 and 15000 as thresholds (40 features) [26];
- Attribute profile of each band with diagonal of bounding box as attribute and 100, 200, 500, 1000 and 20000 as thresholds (40 features) [26];
- Textural features with neighborhood of 19x19 pixels: mean, standard deviation, range and entropy (16 features) [26].

The normalized DSM and the raw RGBIR image are added to these 191 features and then then stacked to create a new image with 196 bands. The training is made with tile 5_11 and test with tile 5_12. Table VI presents the results.

In this case, the classification rate of GMM classifiers with Jeffries-Matusita distance as criterion function drops clearly and has a high standard deviation. It could be due to the fact that the Gaussian hypothesis, on which this metric relies a lot, is not enough verified. When the classification rate is directly used as criterion, the classifier manages to select the relevant features. It has to be noticed that with kappa only 30 features are selected on average when with Jeffries-Matusita distance, the maximum number of features is selected almost always (set to 60).

The KNN classifier and the GMM classifier with ridge regularization are again outperformed even if they get stable results.

Finally, the Random Forest classifier and the GMM with kappa as criterion are from far the best classifiers. The only

TABLE VI
RESULTS OF CLASSIFICATION WITH 1000 SAMPLES BY CLASS AND 5 TRIALS (STANDARD DEVIATION IN PARENTHESIS).

	Kappa of 5_11 (train)	**Kappa of 5_12 (test)**	**Training time (s)**	**Classif. time (s)**	**# features to classify**
GMM SFS kappa	0.694 (0.002)	0.669 (0.005)	400	310	13.2
GMM SFS JM	0.624 (0.028)	0.631 (0.034)	2	310	11
GMM SFFS JM	0.624 (0.028)	0.631 (0.034)	2.6	310	11
GMM ridge	0.632 (0.007)	0.592 (0.010)	10	2000	all
KNN	0.637 (0.005)	0.607 (0.005)			all
Random Forest	0.729 (0.004)	0.673 (0.005)	20	840	all

TABLE VII
RESULTS OF CLASSIFICATION WITH 50000 SAMPLES BY CLASS AND 5 TRIALS (STANDARD DEVIATION IN PARENTHESIS).

	Kappa of 5_11 (train)	**Kappa of 5_12 (test)**	**Training time (s)**	**Classif. time (s)**	**# features to classify**
GMM SFS kappa	0.713 (0.001)	0.684 (0.001)	20000	340	29
GMM SFS JM	0.560 (0.111)	0.576 (0.104)	6	330	10
GMM SFFS JM	0.560 (0.111)	0.576 (0.104)	6.6	340	10
GMM ridge	0.641 (0.015)	0.611 (0.026)	460	2000	all
Random Forest	0.851 (0.001)	0.715 (0.001)	2000	2000	all

difference is in the processing time and also a small difference of classification rate with the training image. Random Forest is faster for the training process while the GMM classifier is faster for classifying.

V. CONCLUSION

An algorithm for the classification of high dimension data as hyperspectral image has been presented. The classifier uses GMM model to select iteratively the most relevant features. Several variation of the algorithms has been explored. Experimentations show that allowing backward step in the selection to discard already selected features do not give significant advantages.

Additionally, from the comparison between all the measures used to rank the features, it has been shown that the Jeffries-Matusita distance can be a fast and accurate solution in some case but suffer from some limitations which remain to be determine. A possible explaination could be that this distance does not perform well when data distribution is not similar to a Gaussian distribution. More investigation is needed to confirm this hypothesis. In any case, it is possible to use directly kappa as criterion function even if it implies a loss of time.

Finally, experiments show that the developed GMM classifier performs at least as best as standard classifiers without features selection in particular Random Forest and even outperforms all of them in term of classification time. The next step would be to compare to other feature selection methods.

The second achievement of this work is the development of an efficient implementation using GMM properties to derive fast update rules of the model. The resulting code is available as a remote module of the Orfeo toolbox on Github and make it possible to process huge quantity of high dimension data.

Finally, an improvement could be made to increase the stability of the selected features. For example, with hyperspectral data, a selection of continuous intervals and not band is a possible solution which has already been explored in [27].

The python and C++ code are available freely for download: https://github.com/Laadr/FFFS, https://github.com/Laadr/otbExternalFastFeaturesSelection.

APPENDIX A

A. Proof of update rules

1) Proposition 4:

Proof.

$$\begin{aligned}\mathbf{A} - \alpha\mathbf{v}\mathbf{v}^t &= (\mathbf{\Sigma}^{(k-1)})^{-1} + \frac{1}{\alpha}(\mathbf{\Sigma}^{(k-1)})^{-1}\mathbf{u}\mathbf{u}^t(\mathbf{\Sigma}^{(k-1)})^{-1} \\ &\quad - \alpha(-\frac{1}{\alpha}(\mathbf{\Sigma}^{(k-1)})^{-1}\mathbf{u})(-\frac{1}{\alpha}\mathbf{u}^t(\mathbf{\Sigma}^{(k-1)})^{-1})) \\ &= (\mathbf{\Sigma}^{(k-1)})^{-1}\end{aligned}$$

□

2) Proposition 5:

Proof.

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

□

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] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE transactions on information theory*, vol. 14, no. 1, pp. 55–63, 1968.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] P. M. Narendra and K. Fukunaga, "A branch and bound algorithm for feature subset selection," *IEEE Transactions on Computers*, vol. 100, no. 9, pp. 917–922, 1977.

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

[13] 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.

[14] 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.

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

[16] 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.

[17] J. Biesiada and W. Duch, "Feature selection for high-dimensional dataa pearson redundancy based filter," in *Computer Recognition Systems 2*. Springer, 2007, pp. 242–249.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] S. Kullback, "Letter to the editor: The kullback-leibler distance," 1987.

[24] 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.

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

PLACE
PHOTO
HERE

Michael Shell Biography text here.

John Doe Biography text here.