




## Open Archive Toulouse Archive Ouverte

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible

This is an author's version published in: <http://oatao.univ-toulouse.fr/21585>

**To cite this version:**

Dechesne, Clément  *Analysis of high dimensional remote sensing images using sparse learning models.* (2014)  
[Mémoire]

Any correspondence concerning this service should be sent to the repository administrator: [tech-oatao@listes-diff.inp-toulouse.fr](mailto:tech-oatao@listes-diff.inp-toulouse.fr)

# Final internship Report

## Analysis of high dimensional remote sensing images using sparse learning models

Clément DECHESNE

3 EN TSI, ENSEEIHT

Supervisor

M. FAUVEL

# Table of Contents

<b>1</b>	<b>State of the art</b>	<b>5</b>
<b>2</b>	<b>Sparse classification methods</b>	<b>6</b>
2.1	Support Vector Machine (SVM)	6
2.1.1	L2 nonlinear	8
2.1.2	Recursive Feature Elimination (RFE)	9
2.1.3	L1 linear	10
2.1.4	Projection in a polynomial space of degree $d$	11
2.2	Nonlinear parsimonious feature selection (NPFS)	13
2.2.1	Gaussian mixture model (GMM)	14
2.2.2	$k$ -fold cross-validation	15
2.2.3	Updating the model	15
2.2.4	Marginalization of Gaussian distribution	16
2.2.5	Leave-one-out cross-validation	17
<b>3</b>	<b>Experimental results</b>	<b>17</b>
3.1	Data set	17
3.2	Results on synthetic data set	18
3.3	Results on real data set	19
3.3.1	Accuracy	20
3.3.2	Selected features	21
3.3.3	Computation time	22
3.3.4	Classification maps	22
<b>4</b>	<b>Leave-One-Out Cross-Validation versus k-fold Cross-Validation</b>	<b>25</b>
4.1	Accuracy	25
4.2	Selected features	26
4.3	Computation time	29
<b>A</b>	<b>Data resizing</b>	<b>33</b>
A.1	No rescale	33
A.2	Rescale between 0 and 1	33
A.3	Rescale between $-1$ and $1$	33
A.4	Standardize	34
A.5	Effect of the resizing	34
<b>B</b>	<b>Simple forward feature selection</b>	<b>35</b>
B.1	Notations	35
B.2	Estimators	36
B.2.1	Proportion	36
B.2.2	Mean vector	36
B.2.3	Covariance matrix	36
B.2.4	Decision rule	36
B.3	Updating formulas	36
B.3.1	Proportion	36
B.3.2	Mean estimation	37
B.3.3	Covariance matrix estimation	38
B.4	LOOCV algorithm	40
<b>C</b>	<b>Figures</b>	<b>41</b>

## Introduction

In the current context of sustainable development, the study of the state and evolution of landscapes is a major challenge to understand and solve environmental problems. The research on the relationship between ecological processes and spatial patterns aims to understand how the configuration and composition of the landscape affect biodiversity.

To address these questions, it is necessary to have a detailed mapping of the elements that make up the landscape. "Detailed" means that it is important to get a map at several levels : for a forest, it is necessary to map the tree species and dead timber on the ground. Another example can be given for the meadows (see figure 1), where the type (permanent, temporary ...) and the composition are spatial variables to extract.



(a) Temporary meadow.



(b) Permanent meadow.

Figure 1: Example of meadows.

Although these spatial data are essential to many socio-ecological models, there is still no mapping for all of these items at local scales. The national mapping data base does not include a detailed "vegetation" layer. The development of this mapping is therefore an important issue for the study of ecosystems.

Multispectral satellite sensors with high spatial resolution allow spatial identification of these objects. But their spectral resolution does not allow a detailed analysis of plant species, phenomenological stages or soil formation. In addition, the low temporal repeatability (revisit time of a site) does not allow a significant inter-annual monitoring.

By their very high spatial and spectral resolution (sampling the spectrum into hundreds of spectral bands), hyperspectral sensors significantly improve the capabilities of remote sensing in this field (see figure 2). It is possible to accurately trace the plant species as well as plant health of semi-natural elements observed. Similarly, the advent of time series of satellite images provides access to a very useful phenomenological information to characterize the response of landscapes to climate change.

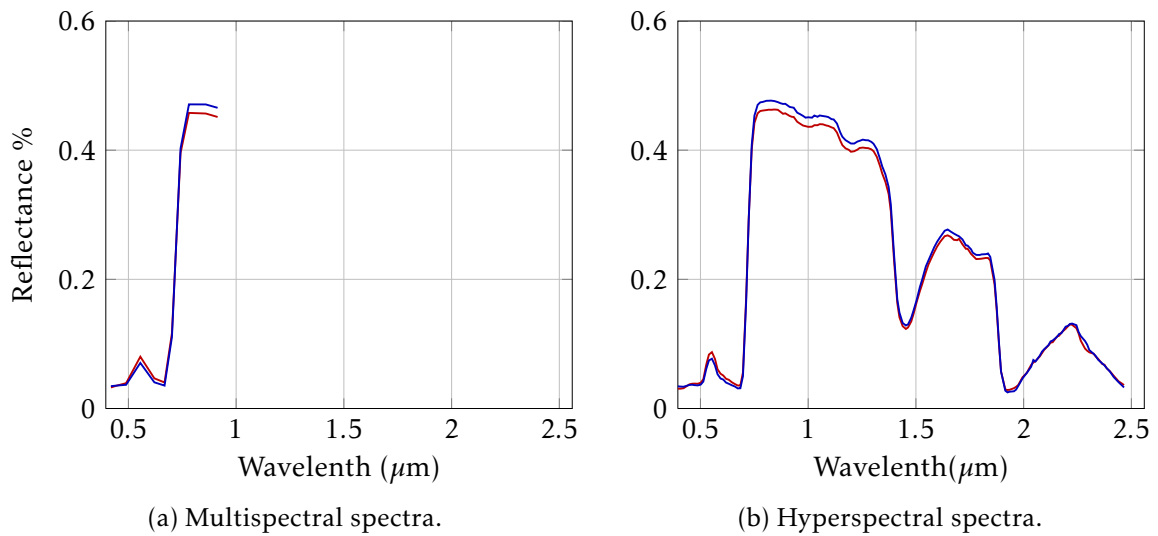


Figure 2: Spectra of two different kind of poplar.

Thus, current images at high spectral and temporal dimension are able to provide valuable information on the status and evolution of landscapes observed (see figure 3). However, the analysis of these images is complicated by the high spectral dimension and/or temporal data. For a pixel, hundreds of spectral variables or tens of time variables are available. There is a strong natural correlation between these variables and therefore redundant information. This redundancy disrupts conventional processing algorithms that have been designed and defined for low dimensional data (few spectral or temporal variables) [1]. This massive flow of information generated necessitates the implementation of intelligent strategies for computer processing.

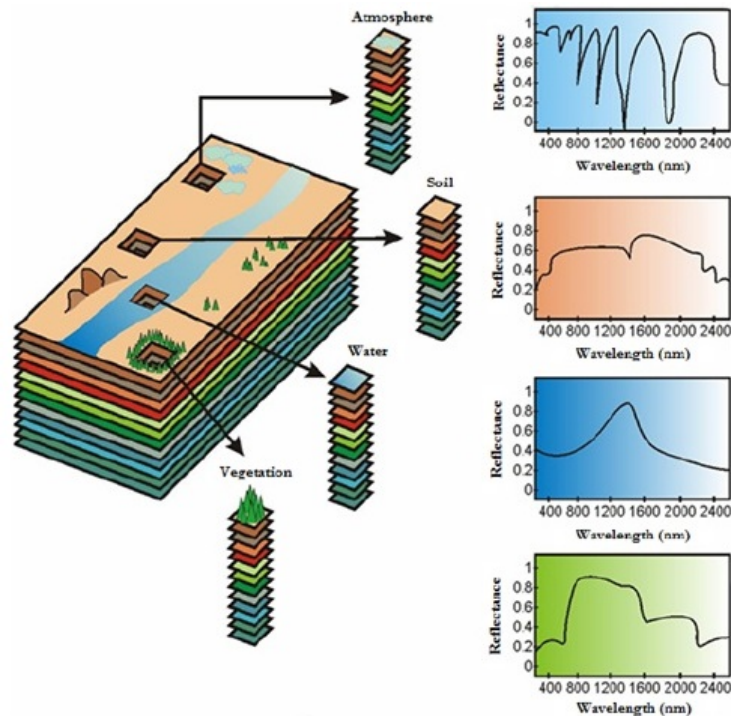


Figure 3: Hyperspectral imaging concept.

Sparse classification methods such as Support Vector Machine (SVM) seem to be appropriate to solve these problems. They are good supervised methods to recognize pattern and it is possible to extend them for feature selection or feature extraction. Some of those extensions are investigated in this internship.

The main goal of this internship was to extend the Leave-One-Out Cross-Validation of the nonlinear parsimonious feature selection [2] presented in section 2.2.5 to a k-fold Cross-Validation (section 2.2.2). The other objective was to compare these new methods, both in term of accuracy and feature selection, to the existing methods, especially linear and nonlinear SVM and Recursive Feature Elimination from nonlinear SVM.

I will first present a state of the art of the feature extraction. The second section present the sparse classifications methods used during my internship; the Support Vector Machine and the Nonlinear Parsimonious Feature Selection. Then a section on synthetic and real data results will be presented. My last point will be a comparison of the leave-one-out cross-validation and the k-fold cross-validation for the Nonlinear Parsimonious Feature Selection.

## 1 State of the art

The reduction of the spectral dimension has received a lot of attention [3]. According to the absence or presence of training set, the dimension reduction can be unsupervised or supervised. The former try to describe the data with a lower number of features that minimize a reconstruction error measure, while the latter try to extract features that maximize the separability of the classes. One of the most used unsupervised feature extraction is the principal component analysis (PCA) [4]. But it has been demonstrated that PCA is not optimal for the purpose of classification [5]. Supervised method, such as the Fisher discriminant analysis or the non-weighted feature extraction have shown to perform better for the purpose of classification. Other feature extraction techniques, such as independent component analysis [6], have been applied successfully and demonstrate that even SVM can benefits from feature reduction [7]. However, conventional supervised techniques suffer from similar problems than classification algorithm in high dimensional space.

Rather than supervised and unsupervised technique, one can also distinguish dimension reduction technique into *feature extraction* and *feature selection*. Feature extraction returns a linear/nonlinear combination of the original features, while feature selection returns a subset of the original features. The latter is much more interpretable for the end-user. The extracted subset corresponds to the most important features for the classification, i.e., the most important wavelengths.

Feature selection techniques generally need a criterion, that evaluates how the model built with a given subset of features performs, and an optimization procedure that tries to find the subset of features that maximizes the criterion [8]. Several methods have been proposed with that setting. For instance, an entropy measure and a genetic algorithm have been proposed in [9, Chapter 9], but the band selection was done independently of the classifier, i.e., the criterion was not directly related to the classification accuracy. Jeffries Matusita distance and forward selection as well as some refinement techniques have been proposed in [8]. However rather than extracting spectral features, the algorithm returns the average over a certain bandwidth of contiguous channels, which can make the interpretation difficult and often leads to select a large part of the spectrum. Recently, forward selection and genetic algorithm driven by the classification error has been proposed in [10].

Alternatively, a shrinkage method based on L1 norm and linear SVM has been investigated Tuia *et al.* [11]. The authors proposed a method where the features are extracted during the training process. However, to make the method tractable in terms of computational load, a linear model is used for the classification which can limit the discriminating

power of the classifier. Feature extraction has been also proposed for non-linear SVM [12] where a recursive scheme is used to remove features that exhibit few influence on the decision function.

During this internship, several feature selection methods were implemented and discussed.

Table 1: Existing methods for feature extraction/selection.

Methods	Reference	Type of the method	Features
PCA	[4]	Unsparse	Extracted
ICA	[6]	Unsparse	Extracted
Spectral interval extraction	[8]	Sparse	Selected
Entropy measure	[9]	Sparse	Selected
Forward selection	[10]	Sparse	Selected
L1 SVM	[11]	Sparse	Selected
RFE SVM	[12]	Sparse	Selected
KPCA	[13]	Unsparse	Extracted
KICA	[14]	Unsparse	Extracted
Fisher discriminant analysis	[15]	Unsparse	Extracted

## 2 Sparse classification methods

This section is divided into two subsections for each of the used methods. The first subsection is about the Support Vector Machine, for which several sparse algorithms are discussed. The second section deals with the Nonlinear Parsimonious Feature Selection, which is a method for feature selection and classification based on Gaussian Mixture Model and developed in this internship.

### 2.1 Support Vector Machine (SVM)

Support vector machines are supervised learning models with associated learning algorithms that analyze data and recognize patterns. Given a set  $\mathcal{S}$  of  $n$  labeled points of the form :

$$\mathcal{S} = \left\{ (\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^L, y_i \in \{-1, 1\} \right\}_{i=1, \dots, n}$$

SVM finds the maximum-margin hyperplane that separate the points having  $y_i = 1$  from those having  $y_i = -1$ . For a pixel  $\mathbf{x}_i$ , the corresponding label can be predicted by  $\hat{y}_i = \text{sign}f(\mathbf{x}_i)$ .  $f$  is such that  $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$ , where  $\mathbf{w}$  is the normal vector to the hyperplane. It has been proved [16] that finding this hyperplane can be done by solving the following equation :

$$\min_f \lambda \sum_{i=1}^n H(y_i, f(\mathbf{x}_i)) + R(f) \quad (1)$$

where  $H$  is called the cost function. For the conventional SVM,  $H$  is the Hinge-Loss function;

$$H(y, f(\mathbf{x})) = \max(1 - yf(\mathbf{x}), 0) = [1 - yf(\mathbf{x})]_+$$

The term  $R$  is the regularization term. In most of the cases,  $R(\cdot) = \frac{1}{2} \|\cdot\|_2^2$  (L2-regularized SVM), it penalizes the large components of the vector  $\mathbf{w}$ . By changing the functions  $L$  and

$R$ , different kind of separating hyperplanes are obtained. The different cost functions are presented in figure 4; the L2-loss function is the one that penalizes the most the wrong classified. The logistic-loss function penalize the less the wrong classified but also penalize the good classified. The hinge-loss function lies between the L2-loss and the logistic-loss. For multiclass data, the separation is made for one class against the others. In that case, (1) is solved  $C$  times (where  $C$  is the number of classes) with  $\forall i \in [1, \dots, n]$ ,  $y_i = 1$  if  $\mathbf{x}_i$  belongs to the class  $c$  ( $c \in [1, \dots, C]$ ) or  $y_i = -1$  if  $\mathbf{x}_i$  does not belong to the class  $c$ .

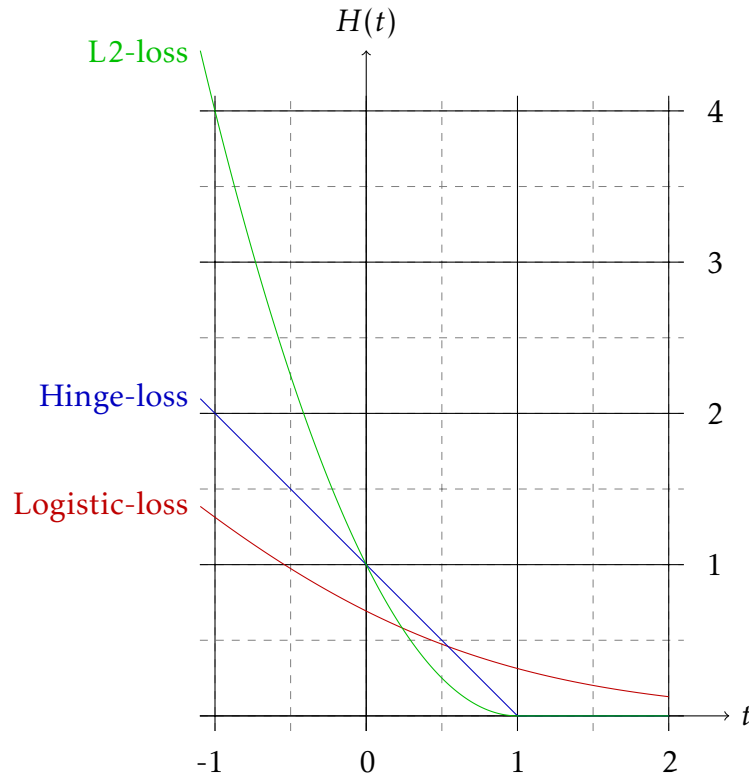


Figure 4: The different cost functions.

Solving these optimization problems allows only a linear separation of the data. To extend the SVM to a nonlinear separation, we can use the kernel trick [16]. This allows to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear and the transformed space high or infinite dimensional; thus though the classifier is a hyperplane in the feature space, it may be nonlinear in the original input space. This method is based on the use of a function  $K$  such as :

$$K(\mathbf{x}_i, \mathbf{x}_j) = (h(\mathbf{x}_i))^T h(\mathbf{x}_j). \quad (2)$$

In the feature space, the vector  $w$  has the following expression [16]  $w = \sum_{i=1}^n \alpha_i y_i h(\mathbf{x}_i)$ . The function  $f$  can be expressed easily with the kernel function  $K$  ;

$$f(\mathbf{x}) = w^T h(\mathbf{x}) + w_0 = \sum_{i=1}^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + w_0.$$

The main used kernels are the following :

– Linear

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^T \mathbf{x}_j. \quad (3)$$



– Radial Basis Function (RBF) or Gaussian

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}. \quad (4)$$

–  $d^{\text{th}}$ -degree polynomial

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 1)^d. \quad (5)$$

To obtain the best accuracy, the optimal parameters need to be founded; the regularization parameter  $\lambda$  and the scale parameter  $\gamma$  or the degree  $d$ . A k-fold cross-validation process has been used and can be summarized as in figure 5. Further details about SVM can be found in [17].

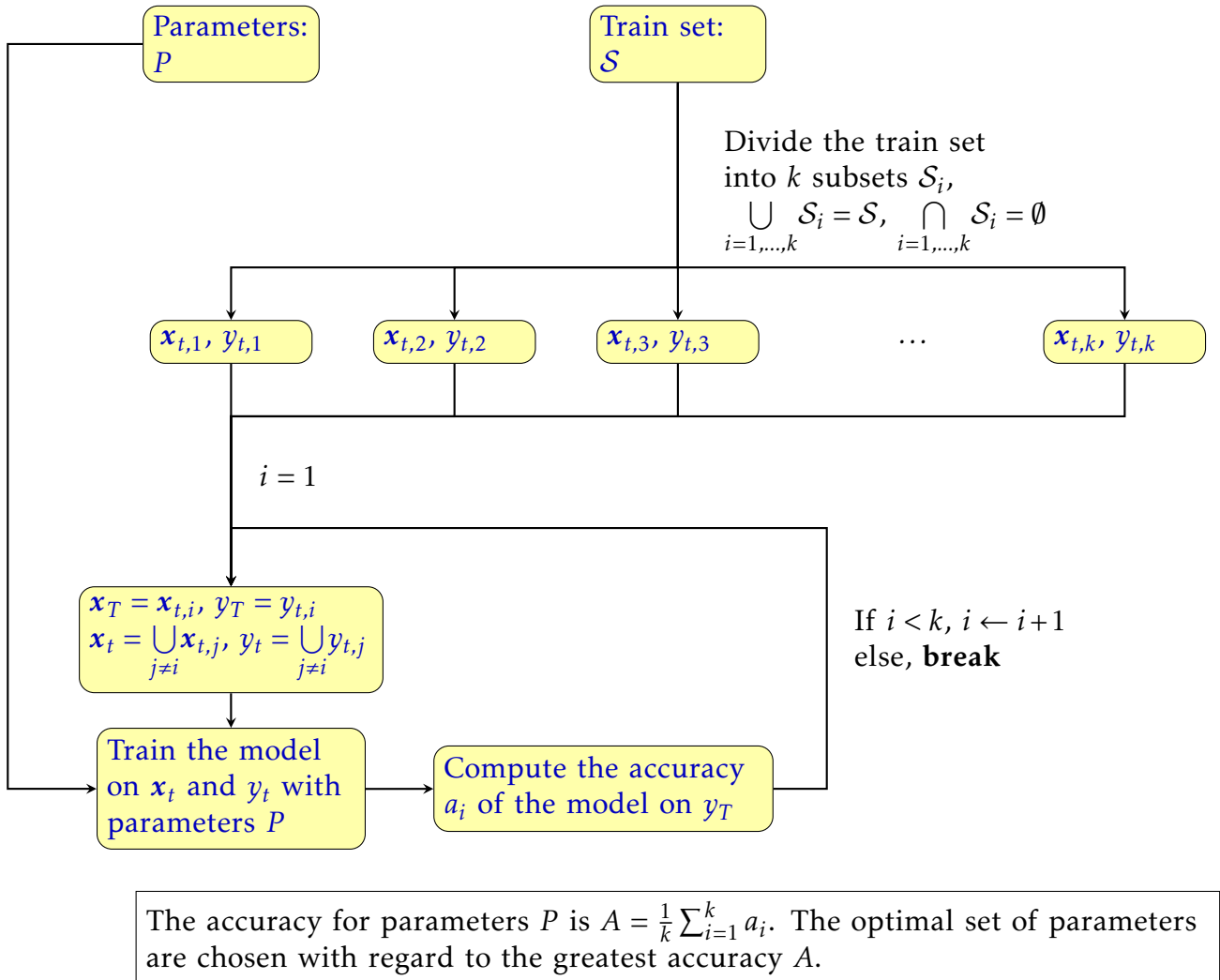


Figure 5: k-fold cross validation method.

### 2.1.1 L2 nonlinear

The non linear L2-regularized SVM are obtained with the RBF or the polynomial kernel, the cost function used is the Hinge-Loss function. We used the library LIBSVM [18] to train and test the model. In non linear cases, it is not possible to compute the normal vector to the hyperplane; one can not find easily the variables which have no influence on the orientation of the hyperplane because the vector  $w$  is in a transformed space. The function  $f$  can be evaluated but it is not possible to determine its parameters ( $w$  and  $w_0$ ). These L2-regularized problems are not the main purpose of the internship because they do not allow

to select explicitly features. However, they will be points of comparison for all the feature selection methods presented in the following sections. Furthermore, the L2-regularized SVM with the RBF kernel is used for the recursive feature selection presented in the next section.

### 2.1.2 Recursive Feature Elimination (RFE)

The L2 nonlinear SVM are good classifiers but do not allow to select variables. However, a method has been proposed [12] to use the nonlinear SVM to select variables. The objective is to remove the variables that will change the less the orientation of the hyperplane while preserving good accuracy rate (see figure 6). Removing variables makes the problem easier to solve, however, this choice is delicate because it can lead to poor results if the wrong variable is removed as it is shown in figure 6.

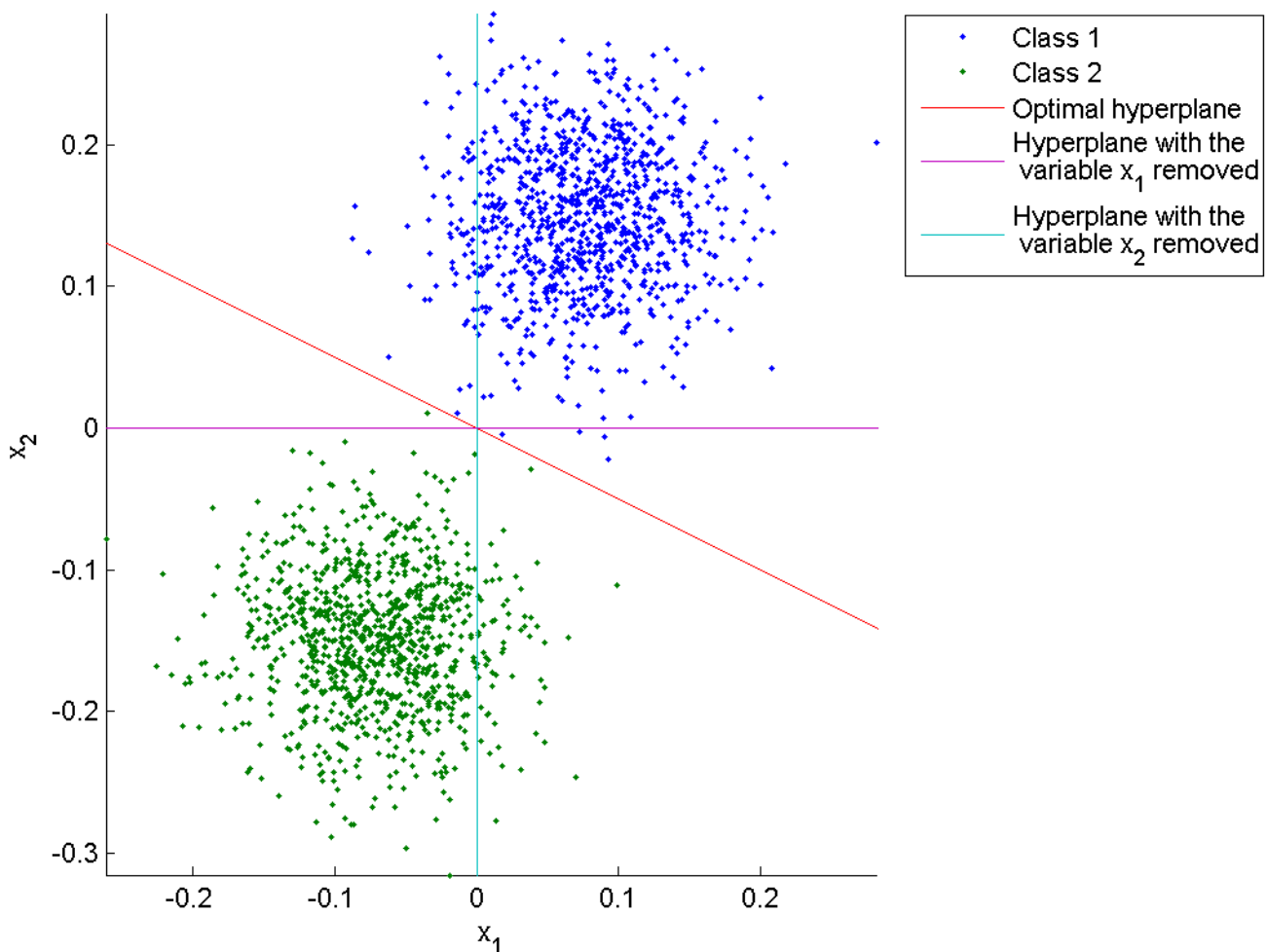


Figure 6: Evolution of the hyperplane when a variables is removed. When the variables  $x_2$  is removed, the separation is bad (94% of good classification), while the removal of the variables  $x_1$  gives a better separation (99.8% of good classification).

The RFE is a method based on nonlinear SVM that allows to remove the less informative feature. The vector  $w$  can not be computed from nonlinear SVM, instead, its norm for the

current hyperplane can be calculated. Here, we will use the square of its norm for the current hyperplane;  $\|w\|^2 = W^2(\alpha)$ .  $\alpha$  denotes the current hyperplane,

$$W^2(\alpha) = \sum_{k,l} \alpha_k \alpha_l y_k y_l K(\mathbf{x}_k, \mathbf{x}_l). \quad (6)$$

This quantity and also the kernel can be easily computed. By using this property and assuming that the  $\alpha_i$  coefficients remain unchanged by removing the less informative feature (i.e. the features that change the less the orientation of the hyperplane), it is possible to compute  $W^2_{(-i)}(\alpha)$  for all the feature subsets counting for all the features minus the considered feature  $i$ . This quantity is computed without retraining the model. Successively, the feature whose removal minimizes the change of the margin is removed, as shown in the following equation:

$$f_m = \arg \min_{i \in [1, \dots, L]} |W^2(\alpha) - W^2_{(-i)}(\alpha)|.$$

To take into account multiclass data, the quantities  $W^2(\alpha)$  and  $W^2_{(-i)}(\alpha)$  are computed separately for each class. The selection criterion is evaluated for the sum over the classes of  $W^2(\alpha)$ ,

$$f_m = \arg \min_{i \in [1, \dots, L]} \sum_{cl} |W^2_{cl}(\alpha) - W^2_{cl,(-i)}(\alpha)|. \quad (7)$$

To use this method, the number of feature to removed need to be specified. In our implementation, we decided to use a threshold  $\delta$ ; when the decrease in term of accuracy after one feature is removed is greater than  $\delta$ , the feature removal stops. An other criterion used is the maximum of variables  $maxvar$  that the algorithm is allowed to remove. If  $maxvar$  variables have been removed, the feature removal stops. The RFE methods is presented in algorithm 1.

---

**Algorithm 1** Recursive Feature Elimination.

---

**Input:**  $S, \delta, maxvar$

$L \leftarrow$  number of bands

$variables \leftarrow [1, \dots, L]$  // Original features

$id \leftarrow []$  // Pool of removed features

$n \leftarrow 1$

Compute the nonlinear SVM model  $m$  and the corresponding accuracy  $a(n)$

**while** 1 **do**

$n \leftarrow n + 1$

    • Get the feature  $f_m$  whose removal minimizes the change of the margin using eq. (7)

    • Remove the feature  $f_m$  from the learning set and from the original set of variables

    • Add the variable  $f_m$  to the pool of removed features

    • Compute the nonlinear SVM model  $m$  and the corresponding accuracy  $a(n)$  using the updated learning set

**if** The decrease in term of  $a > \delta$  or  $\text{length}(id) = maxvar$  **then**

**break**

**end if**

**end while**

---

### 2.1.3 L1 linear

The use of a linear SVM is also interesting because it allows the use of other regularization function more easily; the L1-regularization (i.e.  $R(\cdot) = \|\cdot\|_1$ ) has been widely used for

sparse learning [19]. It has the advantage to provide parsimonious solution; some of the components of  $w$  are exactly zero. The advantage of having a sparse vector  $w$  is that the considered variables  $j$  where  $w_j = 0$  are not relevant in the classification process and can be removed without changing the classification accuracy. We used two different cost function; the Logistic-loss,  $H(y, f(\mathbf{x})) = \log(1 + e^{-yf(\mathbf{x})})$  and the L2-loss,  $H(y, f(\mathbf{x})) = \max(1 - yf(\mathbf{x}), 0)^2 = [1 - yf(\mathbf{x})]_+^2$  (see figure 4). We can see that the Logistic-Loss function is twice differentiable but the L2-Loss function is not. It will lead to higher computation time if we use the L2-Loss function instead of the Logistic-Loss function. To train and test the models, the library LIBLINEAR [20] has been used. It provides a faster and a more complete resolution of the optimization problem than LIBSVM.

#### 2.1.4 Projection in a polynomial space of degree $d$

As it has been explained previously, it could be interesting to project the data into a space where a linear separation correspond to a nonlinear separation in the original feature space. An interesting projective space is a polynomial space because the projection is easy to calculate. When the data are projected, a L1-regularized linear SVM is used. It is more interesting than a L2-regularized SVM with polynomial kernel because the vector  $w$ , which gives the variables or combination of variables characterizing the hyperplane, will be explicitly available. The projection function  $\Phi$  is such that

$$\begin{aligned}\mathbb{R}^L &\rightarrow \mathcal{H} \\ x &\mapsto \Phi(x)\end{aligned}$$

where  $\mathcal{H}$  is the projected space. The dimension of  $\mathcal{H}$  can be easily found;  $\dim(\mathcal{H}) = \binom{L+d}{L}$ . When  $L$  is big (it is the case for hyperspectral images), the dimension of  $\mathcal{H}$  is huge. Therefore, we will use here the projection in a polynomial space of degree 2 to reduce the number of variables in the projected space. The pixel  $\mathbf{x}$  is such as

$$\mathbf{x} = [x_1, x_2, \dots, x_L]^T.$$

The projection  $\Phi(\mathbf{x})$  can be explicited using  $x_1, x_2, \dots, x_L$ :

$$\Phi(\mathbf{x}) = [x_1, x_2, \dots, x_L, x_1^2, \sqrt{2}x_1x_2, \dots, \sqrt{2}x_1x_L, x_2^2, \sqrt{2}x_2x_3, \dots, x_L^2]^T.$$

To illustrate the interest of the projection into a polynomial space of degree 2 and learn a linear sparse model, I will take a toy example with  $x = [x_1, x_2]$  generated from a uniform distribution  $\mathcal{U}(-1, 1)$  and two classes; the class 1 (blue) is attributed to the points where  $x_1^2 + x_2^2 < 0.5$ , while the class 2 (red) attributed to the points where  $x_1^2 + x_2^2 > 0.55$  (see figure 7). The variables needed to separate the two classes linearly,  $x_1^2$  and  $x_2^2$ , are not available in the original feature space.

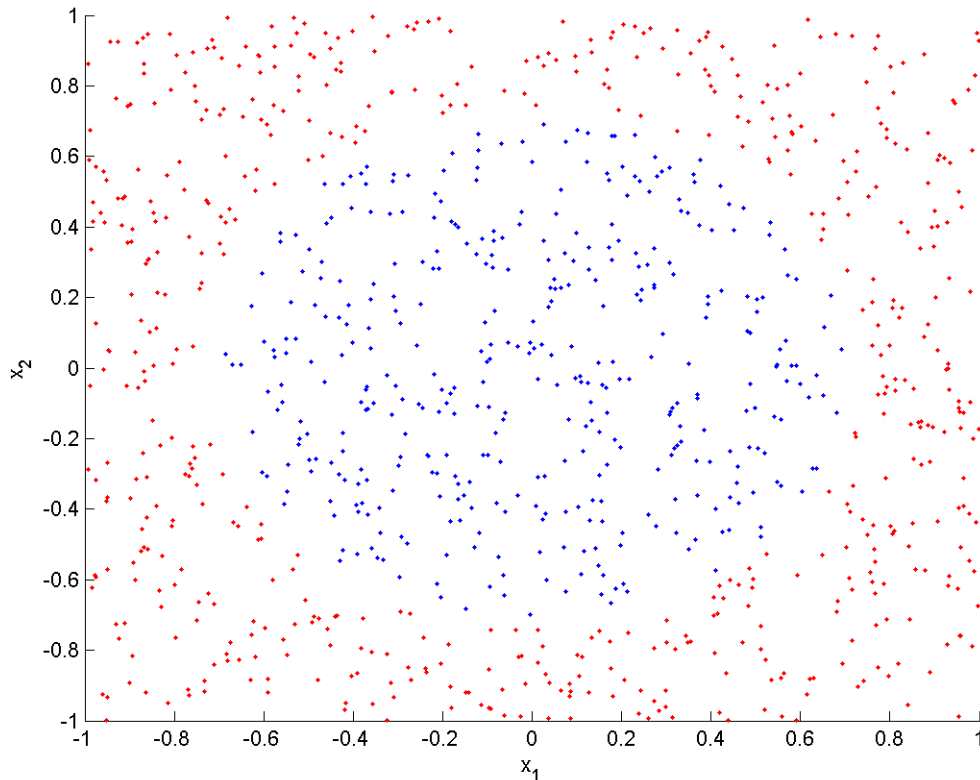


Figure 7: Original data set.

These two class can not be separated using a linear SVM; if we try, the vector  $w$  has no zero components and the correct classification rate is 50.1%. The projection is  $\Phi(x) = [x_1, x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2]$ . On these new variables, a linear separation can be correctly done (see figure 8c) with an accuracy of 100%. The vector  $w$  has two non-zero component corresponding to the variables  $x_1^2$  and  $x_2^2$ , which are the variables that allows to separate the best the two class. This example shows how the projection can be useful; it allows to select not only variables but combination of variable. It also permits to treat data with an accuracy greater than the linear SVM alone. The main problem of the projection is the huge size of the projected space, that may slow the resolution of the equation (1).

*For instance, for the university data set (presented in section 3.1),  $L = 103$  witch leads to  $\dim(\mathcal{H}) = 5460$  for a projection in a polynomial space of degree 2.*

*For the Hekla data set (presented in section 3.1),  $L = 157$  witch leads to  $\dim(\mathcal{H}) = 12560$  for a projection in a polynomial space of degree 2.*

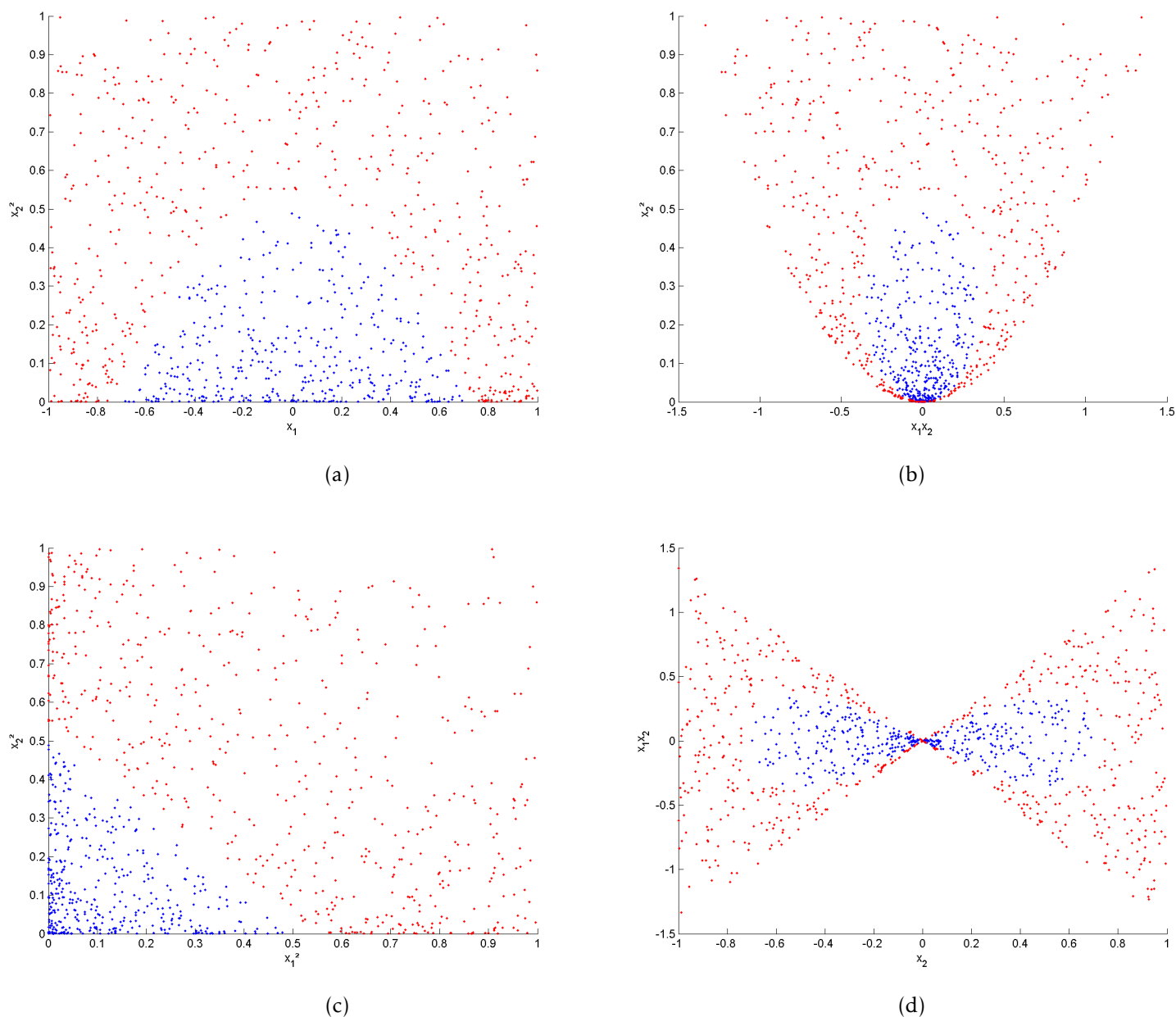


Figure 8: Variables of the projected data set.

## 2.2 Nonlinear parsimonious feature selection (NPFS)

One limitation of the SVM is that it can not do directly multiclass separation and its computation time. Instead, a separation is made between one class and the others for each class. Thus, the interpretation of the selected features is difficult. Furthermore, the feature selection from the SVM are not very efficient (see section 3).

In this section, we propose to use multiclass separation using Gaussian Mixture Model. Therefore, the  $y_i \in [1, \dots, C]_{i=1, \dots, n}$ , are now the label of the class,  $C$  the number of classes and  $n_c$  the number of training pixels in class  $c$  ( $\sum_{c=1}^C n_c = n$ ).

The forward feature selection works as follow. It starts with an empty pool of selected features. At each step, the feature that improves the most the  $k$ -fold cross validation ( $k$ -CV) estimation of the correct classification rate is added to the pool. The algorithm stops either if the increase of the  $k$ -CV is too low or if the maximum number of features is reached. The main contribution of this method is the fast update of the GMM and the fast update using

the marginalization of the GMM.

### 2.2.1 Gaussian mixture model (GMM)

For a Gaussian mixture model, it is suppose that the observed pixel is a realization of a  $L$ -dimensional random vector such as

$$p(\mathbf{x}) = \sum_{c=1}^C \pi_c p(\mathbf{x}|c), \quad (8)$$

where  $\pi_c$  is the proportion of class  $c$  ( $0 \leq \pi_k \leq 1$  and  $\sum_{k=1}^C \pi_k = 1$ ) and  $p(\mathbf{x}|c)$  is a  $L$  dimensional Gaussian distribution.

$$p(\mathbf{x}|c) = \frac{1}{(2\pi)^{L/2} |\Sigma_c|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_c)^T \Sigma_c^{-1} (\mathbf{x}-\boldsymbol{\mu}_c)},$$

where  $\boldsymbol{\mu}_c$  is the mean vector of class  $c$  and  $\Sigma_c$  the covariance matrix of class  $c$ .

Following the maximum a posteriori rule, a given pixel belongs to the class  $c$  if  $p(c|\mathbf{x}) > p(k|\mathbf{x})$  for all  $k = 1, \dots, C$  ( $k \neq c$ ). Using the Bayes formula, the posterior probability can be written as follow

$$p(c|\mathbf{x}) = \frac{\pi_c p(\mathbf{x}|c)}{\sum_{k=1}^C \pi_k p(\mathbf{x}|k)}. \quad (9)$$

Therefore, the maximum a posteriori rule can be written as

$$\mathbf{x} \text{ belong to class } c \iff c = \arg \max_{k=1, \dots, C} \pi_k p(\mathbf{x}|k). \quad (10)$$

By taking the log of the equation (10), the final decision rule is obtained (also known as quadratic discriminant function),

$$Q_k(\mathbf{x}) = -(\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) - \ln(|\Sigma_k|) + 2 \ln(\pi_k). \quad (11)$$

Using standard maximization of the log-likelihood, the estimators of the model are given by

$$\pi_c^n = \frac{n_c}{n}, \quad (12)$$

$$\boldsymbol{\mu}_c^n = \frac{1}{n_c} \sum_{i=1}^{n_c} \mathbf{x}_i, \quad (13)$$

$$\Sigma_c^n = \frac{1}{n_c} \sum_{i=1}^{n_c} (\mathbf{x}_i - \boldsymbol{\mu}_c^n)(\mathbf{x}_i - \boldsymbol{\mu}_c^n)^T. \quad (14)$$

For the GMM, the "Hughes phenomenon" [21] is related to the estimation of the covariance matrix. If the number of training samples is not enough for a good estimation, the computation of the inverse and of the determinant in eq.(11) will be very unstable, leading to poor classification accuracy. For instance for the covariance matrix, that number of parameters to estimate is equal to  $\frac{L \times (L+1)}{2}$ .

For instance, for the University data set (presented in section 3.1),  $L = 103$  then 5356 parameters have to be estimated so the minimum of training samples for the considered class is also 5356.

For the Hekla data set (presented in section 3.1),  $L = 157$  then 12403 parameters have to be estimated so the minimum of training samples for the considered class is also 12403.

Note that in that cases, the estimation will be possible but not accurate.

### 2.2.2 $k$ -fold cross-validation

The efficient implementation of the NPFS relies on the fast estimation of the GMM presented in 2.2.3 and the marginalization of the Gaussian distribution presented in 2.2.4, it is possible to perform a  $k$ -CV and forward selection quickly. The GMM model need to be learned just once during the whole training step. The algorithm 2 present a pseudo code of the proposed method.

---

**Algorithm 2** Simple forward  $k$ -CV feature selection.

---

**Input:**  $S, \delta, \text{maxvariables}, k$   
**Output:** The set of selected variables  $id$   
 $C \leftarrow$  number of classes  
 $n \leftarrow$  number of samples  
 $d \leftarrow$  number of variables  
 $rep \leftarrow 1$   
 $id \leftarrow []$  // Pool of selected features  
 $variable \leftarrow [1, \dots, d]$  // Original features  
 Compute model for each class using eq. (12), (13) and (14)  
 Divide the set  $S$  into  $k$  sets  $S_i$  ( $S = \bigcup_{i=1}^k S_i$ )  
**while**  $rep \leq \text{maxvariable}$  **do**  
    $nv = \text{length}(variable)$  // Number of remaining variables  
    $loocv \leftarrow [0, \dots, 0]$  // Vector of size  $nv$   
   **for**  $i = 1, \dots, k$  **do**  
     Update the model using  $S_i$  as the removed samples  
     **for**  $j = 1, \dots, nv$  **do**  
        $id\_t \leftarrow [id, variable(j)]$   
       Predict the class of the removed samples using the feature selected ( $id\_t$ ) and get the corresponding accuracy  $A$ .  
        $a(j) = a(j) + A$   
     **end for**  
   **end for**  
    $a \leftarrow \frac{a}{k}$   
   Get the maximum of  $a$  and the corresponding variable  $t$   
   **if** The improvement in terms of  $a < \delta$  **then**  
     **break**  
   **else**  
     Add the variable  $t$  to the pool  $id$   
     Remove the variable  $t$  from the original set of variables  
   **end if**  
    $rep \leftarrow rep + 1$   
**end while**

---

### 2.2.3 Updating the model

Here, it is shown that each parameter can be easily updated when a subset of  $S$  is removed.

*Proportion:* The update rule for the proportion is

$$\pi_c^{n-v} = \frac{n\pi_c^n - v_c}{n-v}, \quad (15)$$



where  $\pi_c^{n-v}$  and  $\pi_c^n$  are respectively the proportion of class  $c$  over  $n-v$  and  $n$ ,  $v$  is the number of samples removed from  $\mathcal{S}$ ,  $v_c$  is the number of samples removed from class  $c$  ( $\sum_{k=1}^C v_k = v$ ) (see appendix B.3.1 for proofs).

*Mean vector:* The update rule for the mean vector is

$$\mu_c^{n-v_c} = \frac{n_c \mu_c^{n_c} - v_c \mu_c^{v_c}}{n_c - v_c}, \quad (16)$$

where  $\mu_c^{n_c-v_c}$  and  $\mu_c^{n_c}$  are respectively the mean vectors of class  $c$  computed over the  $n_c - v_c$  and  $n_c$  training samples,  $\mu_c^{v_c}$  is the mean vector of the  $v_c$  samples removed from class  $c$  (see appendix B.3.2 for proofs).

*Covariance matrix:* The update rule for the covariance matrix is

$$\Sigma_c^{n_c-v_c} = \frac{n_c}{n_c - v_c} \Sigma_c^{n_c} - \frac{v_c}{n_c - v_c} \Sigma_c^{v_c} - \frac{n_c v_c}{(n_c - v_c)^2} (\mu_c^{v_c} - \mu_c^{n_c})(\mu_c^{v_c} - \mu_c^{n_c})^T, \quad (17)$$

where  $\Sigma_c^{n_c-v_c}$  and  $\Sigma_c^{n_c}$  are respectively the covariance matrix of class  $c$  computed over the  $n_c - v_c$  and  $n_c$  training samples,  $\Sigma_c^{v_c}$  is the covariance matrix of the  $v_c$  samples removed from class  $c$  (see appendix B.3.3 for proofs).

The table 2 allows to compare the difference in term of computation time between recalculating the model and updating the model (the percentage expresses the rapidity compared to the other method). We can note that the error made are more numerical errors and will not have a big impact on the classification. In most of the cases, updating the model is at least 70% faster than re-training the model. In our case, the model need to be updated  $k \times nv$  (where  $nv$  is the number of remaining variables). On hyperspectral images, where the number of variables is important, this gain of time can lead to substantial computation time if the model was re-computed instead of updated. This gain of time is much more important when a leave-one-out cross validation (section 2.2.5) is done (in that case,  $k = n$ ).

Table 2: Computation time comparison, a fifth of the samples has been removed ( $L = 200$ , 8 classes).

Number of samples	Re-compute the model	Update the model	Mean error on the covariance matrix
800 (160 removed)	<b>3.22e-03 s, (56.4 %)</b>	7.39e-03 s	4.22e-17
4000 (800 removed)	1.01e-02 s	<b>5.96e-03 s, (40.7 %)</b>	4.48e-17
8000 (1600 removed)	2.85e-02 s	<b>7.17e-03 s, (74.8 %)</b>	4.57e-17
40000 (8000 removed)	1.44e-01 s	<b>3.14e-02 s, (78.2 %)</b>	5.19e-17
80000 (16000 removed)	2.76e-01 s	<b>5.01e-02 s, (81.8 %)</b>	6.37e-17
800000 (160000 removed)	2.78e+00 s	<b>5.13e-01 s, (81.5 %)</b>	1.30e-16
42776 (8555 removed) (University data set)	6.79e-02 s	<b>1.35e-02 s, (80.1 %)</b>	1.13e-11
10227 (2045 removed) (Hekla data set)	2.69e-02 s	<b>8.50e-03 s, (68.4 %)</b>	2.69e-13

## 2.2.4 Marginalization of Gaussian distribution

To get the GMM over a subset of the original set of feature (i.e. the pool of selected feature), it is only necessary to drop the non selected features from the mean vector and the

covariance matrix [22]. for instance, let  $\mathbf{x} = [\mathbf{x}_s, \mathbf{x}_{ns}]$ , where  $\mathbf{x}_s$  and  $\mathbf{x}_{ns}$  are respectively the selected and the non-selected variables. The mean vector and the covariance matrix can be written as :

$$\boldsymbol{\mu} = [\boldsymbol{\mu}_s, \boldsymbol{\mu}_{ns}]^T, \quad (18)$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{s,s} & \boldsymbol{\Sigma}_{s,ns} \\ \boldsymbol{\Sigma}_{ns,s} & \boldsymbol{\Sigma}_{ns,ns} \end{bmatrix}. \quad (19)$$

The marginalization over the non-selected variables shows that  $\mathbf{x}_s$  is also a Gaussian distribution with mean vector  $\boldsymbol{\mu}_s$  and covariance matrix  $\boldsymbol{\Sigma}_{s,s}$ . Hence, once the full model is learned, all the sub-models built with a subset of the original variables are available at no computational cost.

### 2.2.5 Leave-one-out cross-validation

When a few training samples are available, it is wiser to resort to leave-one-out cross validation (only one sample  $\mathbf{x}_n$  is removed from  $\mathcal{S}$ ). The update rules are still valid and easier to write ( $v = v_c = 1$ ,  $\boldsymbol{\mu}_c^{v_c} = \mathbf{x}_n$ ,  $\boldsymbol{\Sigma}_c^{v_c} = 0$ ). It is also possible to get a fast update of the decision function when the removed sample does not belong to the class  $c$ ; the only changing parameter in equation (11) is the proportion. Therefore, the update of the decision rule can be written as follows

$$Q_c^{n_c-1}(\mathbf{x}_n) = Q_c^{n_c}(\mathbf{x}_n) + 2 \ln\left(\frac{n-1}{n}\right). \quad (20)$$

An update rule for the case where the sample belongs to the class  $c$  can be written by using the Cholesky decomposition of the covariance matrix and rank-one downdate, but the downdate is not numerically stable and not used here. A pseudo code for the LOOCV is detailed in algorithm 3 in appendix B.4.

## 3 Experimental results

### 3.1 Data set

For the test of the efficiency of the algorithms, synthetic data were generated. A noise from a uniform distribution is generated over the  $L$  bands and the classes are assigned based on the value of the variables for each sample. The separation for each variables is linear, but the separations of the class is not linear but just piecewise linear. The synthetic data are very useful because they help to prove that the selected variables from the different methods are effectively the variables of interest defined by the user. An example of synthetic data set is presented in figure 9.

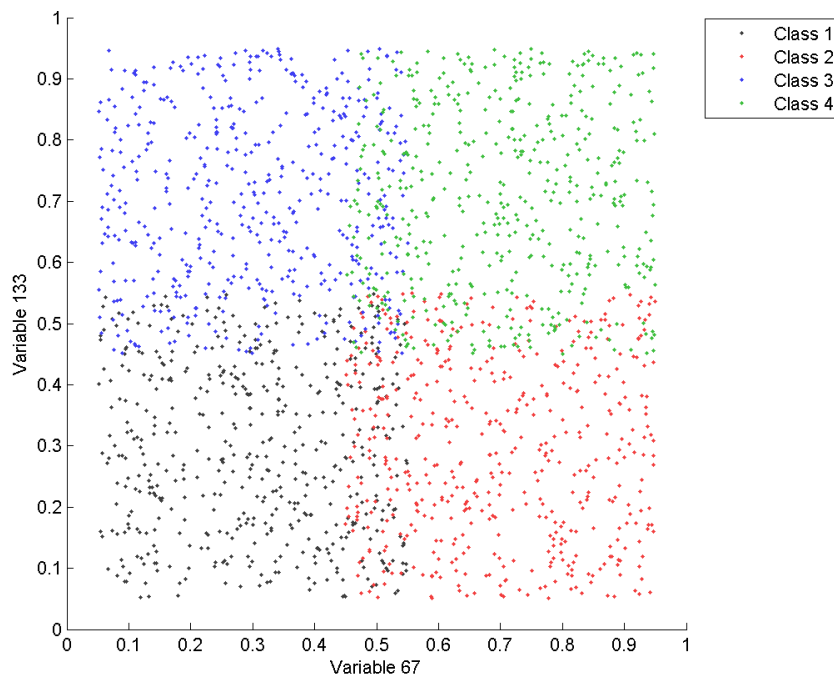


Figure 9: Synthetic data with 2 discriminant features over the 200 available.

Two real data sets have been used in the experiments. The first data set has been acquired by the ROSIS sensor during a flight campaign over Pavia, northern Italy;

- 103 spectral channels from 430nm to 860nm recorded,
- 9 classes defined,
- 42776 referenced pixels.

The second data set has been acquired in the region surrounding the volcano Hekla in Iceland by the AVIRIS sensor;

- 157 spectral channels from 400nm to 1840nm recorded,
- 12 classes defined,
- 10227 referenced pixels.

For each data set, 50, 100 and 200 training pixels per class were randomly selected and the remaining referenced pixels were used for the validation. 50 repetitions were done for which a new training set have been generated. All the  $k$ -fold cross-validation were done with  $k = 5$ .

### 3.2 Results on synthetic data set

- Feature selected  
The table 3 shows that the simple forward algorithms are very efficient because they select 3, 4 or at most 5 features over the 200 available. In the pool of the selected features, there are always the discriminant ones. When a lot of samples are available, the simple forward algorithms always select only the discriminant features.

The linear SVM also select the discriminant features. However, the number of selected features is much more important than the simple forward algorithms. This is due to the the method of multiclass classification; for each class, a lot of different features are selected. However, the discriminant features are always selected for all the classes.

Table 3: Number of selected features for synthetic data with 3 discriminant features over the 200 available (8 classes).

	$n_c = 50$	$n_c = 100$	$n_c = 200$
LIBLINEAR: L1-regularized, L2-loss	$7.9 \pm 0.6$	$10.2 \pm 1.1$	$13.6 \pm 1.8$
Simple forward LOOCV ( $\delta = 0.50$ )	$3.2 \pm 0.2$	$3.2 \pm 0.2$	$3.0 \pm 0.0$
Simple forward k-CV ( $\delta = 0.50$ )	$3.5 \pm 0.3$	$3.1 \pm 0.1$	$3.0 \pm 0.0$

- Accuracy

The tables 4 shows an other advantage of the simple forward algorithms; their accuracy is better by at least 15% than the L1-regularized linear SVM. However, these synthetic data are not really linearly separable in one versus all multiclassification (see figure 9). It explain the poor results of the linear SVM. Furthermore, the improvement in term of accuracy for the linear SVM is of about 3% between  $n_c = 50$  and  $n_c = 200$  while the improvement for the simple forward algorithms is at least of 5%.

Table 4: Accuracy for synthetic data with 3 discriminant features over the 200 available (8 classes).

	$n_c = 50$	$n_c = 100$	$n_c = 200$
LIBLINEAR: L1-regularized, L2-loss	$69.2 \pm 3.7$	$71.9 \pm 1.4$	$72.8 \pm 0.4$
Simple forward LOOCV ( $\delta = 0.50$ )	$84.9 \pm 4.0$	$88.2 \pm 1.2$	$90.4 \pm 0.5$
Simple forward k-CV ( $\delta = 0.50$ )	$84.0 \pm 5.5$	$88.2 \pm 1.3$	$90.2 \pm 0.8$

- computation time

Also in term of computation time, the forward algorithms are faster (some seconds) compared to the L1-regularized linear SVM (about 15 minutes).

The results on synthetic data allows to point out some big advantages of the simple forward algorithms; they select few features and always the good ones, they have a good accuracy for short computation time. However, even if the linear SVM has a bad classification accuracy, it also select the good features with reasonable computation time.

### 3.3 Results on real data set

In this section, the following color code is used; blue correspond to L1-regularized linear SVM, black correspond to the SVM with polynomial kernel or the L1-regularized linear SVM on the data projected in the polynomial space of degree 2, green correspond to the nonlinear SVM with gaussian kernel and red correspond to the simple forward algorithms. The best result for each training setup is reported in bold face.

### 3.3.1 Accuracy

Table 5: Mean and variance of the accuracy for the proposed methods.

	$n_c = 50$	$n_c = 100$	$n_c = 200$
<b>University</b>			
LIBLINEAR: L1-regularized, Logistic-loss	75.1 ± 2.5	77.3 ± 1.4	78.5 ± 0.7
LIBLINEAR: L1-regularized, L2-loss	74.8 ± 2.6	76.9 ± 1.9	78.2 ± 0.8
LIBLINEAR: L1-regularized, Logistic-loss, projected	81.0 ± 2.8	83.6 ± 1.3	85.5 ± 0.4
LIBSVM: RBF kernel	82.9 ± 3.4	86.5 ± 1.6	88.8 ± 0.6
LIBSVM: RBF kernel	84.8 ± 3.4	88.4 ± 1.4	90.8 ± 0.3
RFE (RBF kernel, $\delta = 0.10$ )	84.7 ± 4.0	88.4 ± 0.9	90.8 ± 0.3
Simple forward LOOCV ( $\delta = 0.50$ )	84.2 ± 4.4	86.3 ± 3.2	87.7 ± 3.1
Simple forward k-CV ( $\delta = 0.50$ )	83.4 ± 7.6	85.9 ± 3.1	87.9 ± 1.9
<b>Hekla</b>			
LIBLINEAR: L1-regularized, Logistic-loss	90.3 ± 1.0	93.9 ± 0.5	95.6 ± 0.1
LIBLINEAR: L1-regularized, L2-loss	89.9 ± 1.2	94.4 ± 0.2	96.4 ± 0.3
LIBLINEAR: L1-regularized, Logistic-loss, projected	91.6 ± 0.6	94.8 ± 0.1	96.3 ± 0.1
LIBSVM: Polynomial kernel	84.6 ± 1.6	91.4 ± 0.4	95.5 ± 0.1
LIBSVM: RBF kernel	90.4 ± 1.6	95.6 ± 0.3	96.8 ± 1.1
RFE (RBF kernel, $\delta = 0.10$ )	90.2 ± 1.8	95.6 ± 0.3	96.8 ± 1.1
Simple forward LOOCV ( $\delta = 0.50$ )	92.5 ± 1.2	94.8 ± 0.7	95.9 ± 0.3
Simple forward k-CV ( $\delta = 0.50$ )	92.4 ± 1.2	94.6 ± 0.6	95.8 ± 0.3

#### • University data set

We can see that the linear SVM has the worst results. However, the Logistic-loss linear SVM has slightly better results than the L2-loss linear SVM.

The accuracy of the SVM with polynomial kernel is slightly better (2%-3%) than the accuracy on the projected data with a linear SVM. The accuracy of these methods are still better than the accuracy of the linear SVM on original data.

The simple forward algorithms and the SVM with RBF kernel have similar results. The SVM with RBF kernel has the greatest accuracy (with or without RFE) when the simple forward algorithms are slightly worse. We can note that the difference between the SVM with RBF kernel and the simple forward algorithms is lower than 3%. When the number of samples per class increase, the accuracy of the simple forward algorithms does not increase a lot (2%-3%) while the accuracy of the nonlinear SVM increase significantly (6%-8%).

#### • Hekla data set

The different methods obtained similar accuracy; the greatest accuracy is reached by the simple forward algorithms for  $n_c = 50$  and by the SVM with RBF kernel for  $n_c = 100$  and  $n_c = 200$ . The worst results are obtained with the SVM with polynomial kernel.

The good accuracy of the linear SVM suggest that these data set can be easily linearly separable. It also explains the results of the linear SVM on projected data, which is better than the SVM with polynomial kernel.

The linear SVM performs slightly worse in term of accuracy than the SVM with RBF kernel or the simple forward algorithms.

We can note that the difference between the SVM with RBF kernel and the simple forward algorithms is lower than 2%. When the number of samples per class increase, the accuracy

of the simple forward algorithms does not increase a lot (3%-4%) while the accuracy of the nonlinear SVM increase significantly (6%-7%).

In term of accuracy, the nonlinear SVM with RBF kernel has in general the greatest accuracy. However, the simple forward algorithms are also very efficient. The efficiency of the L1-regularized linear SVM depends on the data (the Hekla data set is more linearly separable than the University data set) and does not have the best results. The projection in the polynomial space has better results than the L1-regularized SVM on original data, but it is still worse than the nonlinear SVM with RBF kernel or the simple forward algorithms.

### 3.3.2 Selected features

Table 6: Mean and variance of the number of selected features for the proposed methods.

	$n_c = 50$	$n_c = 100$	$n_c = 200$
<b>University</b>			
LIBLINEAR: L1-regularized, Logistic-loss	61.6 ± 5.1	78.3 ± 5.1	92.3 ± 2.7
LIBLINEAR: L1-regularized, L2-loss	42.0 ± 3.3	44.0 ± 2.1	56.2 ± 2.8
LIBLINEAR: L1-regularized, Logistic-loss projected	52.0% ± 29.4%	56.7% ± 25.8%	76.2% ± 24.9%
RFE (RBF kernel, $\delta = 0.10$ )	98.4 ± 54.3	99.1 ± 12.8	99.3 ± 12.8
Simple forward LOOCV ( $\delta = 0.50$ )	6.4 ± 1.3	6.5 ± 0.7	6.9 ± 1.2
Simple forward k-CV ( $\delta = 0.50$ )	6.1 ± 1.0	6.6 ± 1.0	7.0 ± 0.8
<b>Hekla</b>			
LIBLINEAR: L1-regularized, Logistic-loss	157.0 ± 0.0	157.0 ± 0.0	157.0 ± 0.0
LIBLINEAR: L1-regularized, L2-loss	157.0 ± 0.0	157.0 ± 0.0	157.0 ± 0.0
LIBLINEAR: L1-regularized, Logistic-loss projected	93.6% ± 14.0%	91.7% ± 17.1%	92.7% ± 15.8%
RFE (RBF kernel, $\delta = 0.10$ )	154.7 ± 4.2	155.2 ± 1.8	155.3 ± 1.2
Simple forward LOOCV ( $\delta = 0.50$ )	7.6 ± 1.7	8.5 ± 1.4	8.8 ± 0.7
Simple forward k-CV ( $\delta = 0.50$ )	7.3 ± 1.0	8.2 ± 1.1	8.6 ± 0.9

#### • University data set

The simple forward methods select a few variables (6 or 7); it is less than 7% of the total of the original features.

The linear SVM selects a lot of variables (about 40% of all the variables for the method which select the less features). It is interesting to note the difference in the number of selected features between the L2-loss and the Logistic-loss; the L2-loss selects fewer variables. This is due to the greater weight accorded to a missclassification (see figure 4); the regularization parameter  $\lambda$  can be small, ensuring a great sparseness (see figures 22, 23 and 24).

The RFE does not remove a lot of variables (4 or 5 variables removed).

The linear SVM on projected data selects 52% to 76% of the projected variables (5460 variables) which is more than the original number of variables. The original features are nearly always selected and many different combination of features are selected.

#### • Hekla data set

The simple forward also selects a few variables (7 or 8); it is less than 6% of the total of the original features.

The RFE removes only some variables (2 or 3 variables removed).

The linear SVM does not select any variables. If we look to the figure 25, 26 and 27 in appendix C, we can see that if we accept to reduce the accuracy, the linear SVM would be able to select from 70% to 80% of the total number of variables (there would be 20% to 30% of

zeros in the vector  $w$ ). It is still too much compared to the number of variables selected by the simple forward algorithms.

The linear SVM on projected data extracts 92% to 93% of the projected variables (12560 variables) which is still more than the original number of variables. On this image, the selected features are equally spread.

The simple forward algorithms seem to be the more appropriate in term of number of selected features. The L1-regularized linear SVM can select features, but the number of features selected highly depends on the data; for one set the L1-regularized linear SVM was able to select features while it was not for the other set. The RFE is not relevant for feature selection, only a few features are removed from the set; it is as if no features have been selected. The features extracted with the L1-regularized SVM on projected data are not interesting; the number of extracted features is greater than the original number of features.

### 3.3.3 Computation time

#### • University data set

In term of computation time, the simple forward algorithms are the fastest (some seconds). The linear SVM is also fast (some minutes). The nonlinear SVM comes just after with substantial computation time ( $\sim 15$  minutes). The computation time for the RFE is much worse (some hours). The worst computation time is obtained on the projected data using the linear SVM (about 50 hours for  $n_c = 200$ ).

#### • Hekla data set

The computation time are in general slightly longer than the one obtained on the University data set. This is due to the greater number of spectral variables and number of samples. For the linear SVM on projected data, computation time are much more longer than the one observed on the University data set (about 160 hours for  $n_c = 200$ ).

### 3.3.4 Classification maps










The different methods have been applied on real hyperspectral images. In these application, the spatial information has not been taken into account; each pixel has been treated without regard to the value of its neighbors.

To reduce computation time all the methods were trained with 500 samples per class for the University data set and 300 samples per class for the Hekla data set.













The University of Pavia's maps results are presented in figure 10. The Hekla's maps results are presented in figure 11. These maps do not allow to see the difference between the used methods, especially for the simple forward algorithms. Nevertheless, they reflect the results presented in section 3.3.1; the SVM has the greatest results, the simple forward algorithms perform slightly worse in term of accuracy. For the University data set, the linear SVM has poor results. For the Hekla data set, the linear SVM has similar results to other methods.

The absence of spacial information makes the maps very noisy. For the simple forward algorithms, the probability of belonging to a class is available, and the implementation of a Markov chain could be possible and would make the maps more reliable.

**Groundtruth classes for the Pavia University scene and their respective samples number**

	Class	Samples
	Asphalt	6631
	Meadows	18649
	Gravel	2099
	Trees	3064
	Painted metal sheets	1345
	Bare Soil	5029
	Bitumen	1330
	Self-Blocking Bricks	3682
	Shadows	947

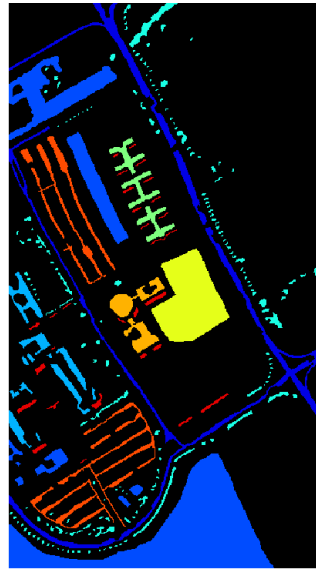
**Groundtruth classes for the Hekla scene and their respective samples number**

	Class	Samples
	Hyaloclastite formation	342
	Tephra lava	404
	Lava 1980 II	410
	Snow	458
	Lava 1991 I	550
	Lava 1980 I	684
	Lava moss cover	700
	Rhyolite	708
	Scoria	713
	Lava 1970	1023
	Lava 1991 II	1496
	Firn-glacier ice	2739

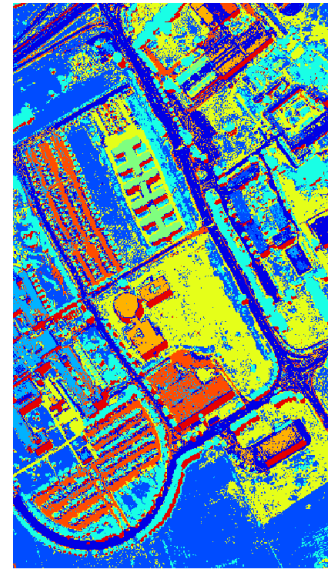




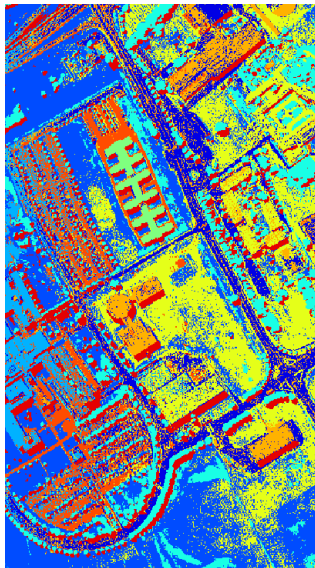
(a) RGB colored image of the University of Pavia



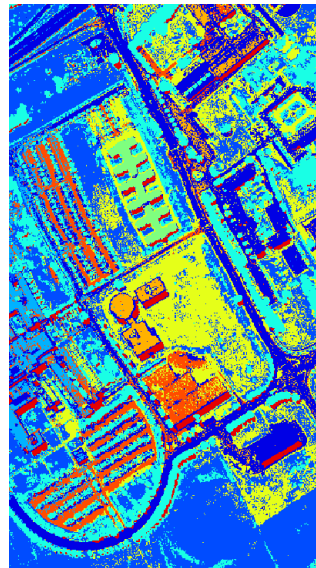
(b) Ground truth of the University of Pavia



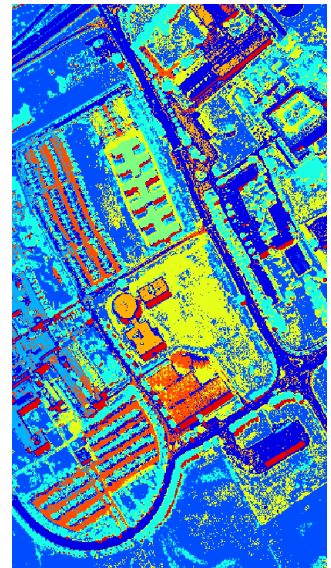
(c) Map estimated with a SVM using a RBF kernel



(d) Map estimated with a linear SVM



(e) Map estimated with the simple forward LOOCV



(f) Map estimated with the simple forward k-CV

Figure 10: Maps obtained with the different methods on the University scene.

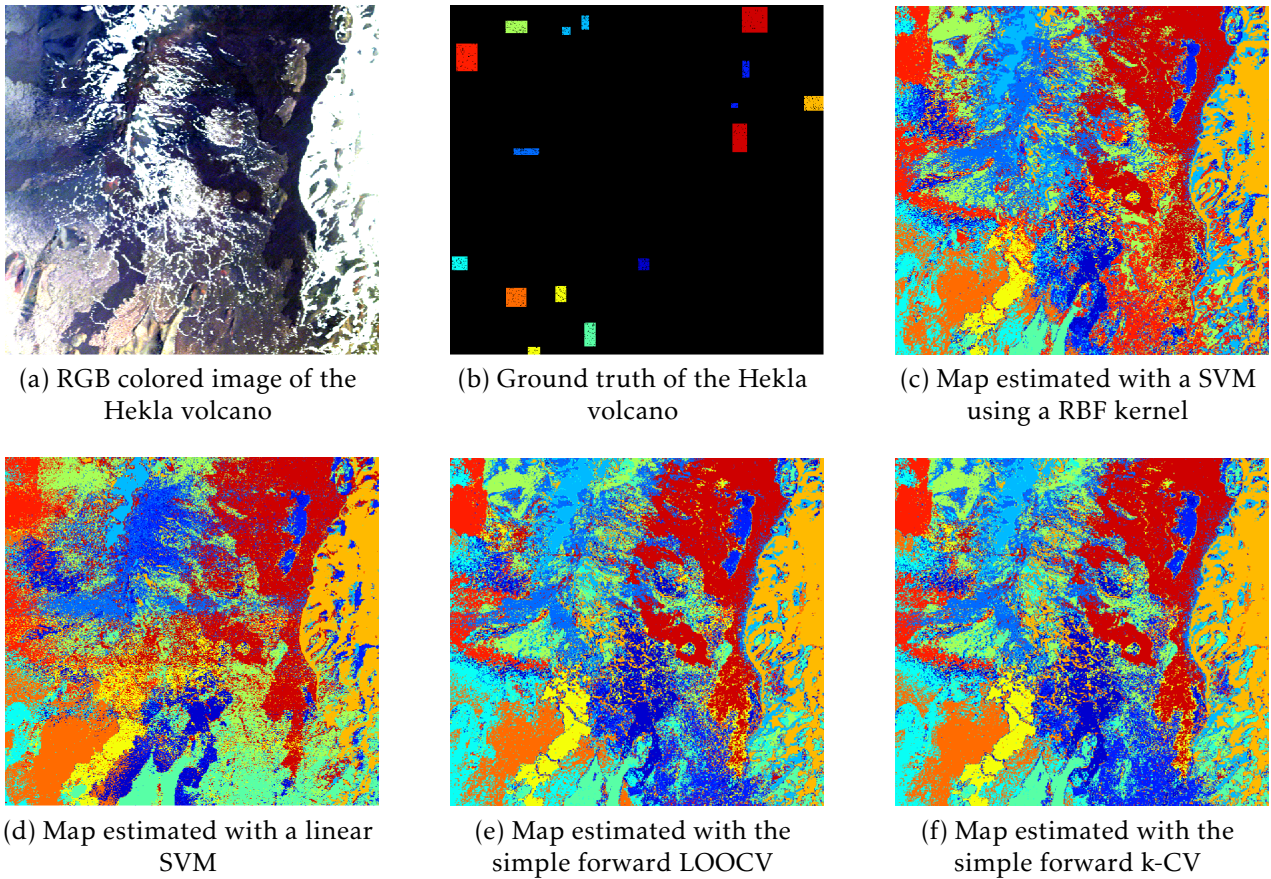


Figure 11: Maps obtained with the different methods on the Hekla scene.

## 4 Leave-One-Out Cross-Validation versus k-fold Cross-Validation

### 4.1 Accuracy

In term of accuracy, the two algorithms are similar. However, the LOOCV work better than the k-CV when few samples are available. Indeed, in that case, the covariance matrices of the GMM might be ill-conditioned and it occurs numerical errors on eq.(11). The LOOCV will lead to small change on the covariance matrix while the k-CV might exacerbate the ill-conditioning of the covariance matrix.

When a lot of samples are available, the k-CV is a bit more efficient than the LOOCV; only one sample is removed from a large number of samples (when  $n_c$  is large), the parameters of the GMM are not a lot changed;

$$\begin{aligned}\pi_c^{n-1} &= \frac{n\pi_c^n - 1}{n-1} \approx \pi_c^n, \\ \mu_c^{n_c-1} &= \frac{n_c \mu_c^{n_c} - \mathbf{x}_n^{v_c}}{n_c - 1} \approx \mu_c^{n_c}, \\ \Sigma_c^{n_c-1} &= \frac{n_c}{n_c - 1} \Sigma_c^{n_c} - \frac{n_c}{(n_c - 1)^2} (\mathbf{x}_n - \mu_c^{n_c})(\mathbf{x}_n - \mu_c^{n_c})^T \approx \Sigma_c^{n_c}, \\ Q_c^{n_c-1}(\mathbf{x}_n) &= Q_c^{n_c}(\mathbf{x}_n) + 2 \ln\left(\frac{n-1}{n}\right) \approx Q_c^{n_c}(\mathbf{x}_n).\end{aligned}$$

In that case, the k-CV will occur more variability while the LOOCV would have the same effect as marginalize the original GMM.

## 4.2 Selected features

In term of selected features, the two algorithms are also similar. For a given training set of the University of Pavia with  $n_c = 200$ , 6 features are selected with the k-CV and 7 features are selected with the LOOCV (see figure 12). On this image, a channel width is about 4nm. From these selected features, three were the same for both methods; 607nm, 780nm and 784nm. Two channel were very close, 548nm and 725nm for the k-CV and 544nm and 721nm for the LOOCV. One was close, 485nm for the k-CV and 510nm for the LOOCV. The last feature selected by the LOOCV is at 704nm, which is also close to the feature at 725nm for the k-CV and 721nm for the LOOCV.

If the process is repeated, the result in term of selected features by the two methods is similar; about 35% of the selected features are identical and the others selected features are close in term of wavelenghts.

For a given training set of Hekla with  $n_c = 200$ , 9 features are selected with the k-CV and 9 features are selected with the LOOCV (see figure 13). On this image, a channel width is about 9nm. From these selected features, four were the same for both methods; 483.1nm, 621.5nm, 990.8nm and 1507.7nm . Two channel were very close, 446.2nm and 501.5nm for the k-CV and 455.4nm and 492.3nm for the LOOCV. Three were close, 880.0nm, 1092.3nm and 1673.8nm for the k-CV and 926.2nm, 1064.6nm and 1590.8nm for the LOOCV.

If the process is repeated, the result in term of selected features by the two methods is similar; about 40% of the selected features are identical and the others selected features are close in term of wavelenghts.

The table 7 shows the number of features selected by both the k-CV and the LOOCV for a given set over 50 repetitions.

The figures 14 present the most selected features for the University of Pavia data set. 1000 random repetitions have been done with  $n_c = 200$  and the features shaded in the figures have been selected at least 10% time times using k-CV. The width of the spectral domain indicates the variability of the selection. The high correlation between adjacent bands makes the bands selection "unstable"; for a given training set, the band  $b$  would be selected, but for another randomly selected training set, it might be the band  $b + 1$  or  $b - 1$ . It is a limitation of the simple forward algorithms.

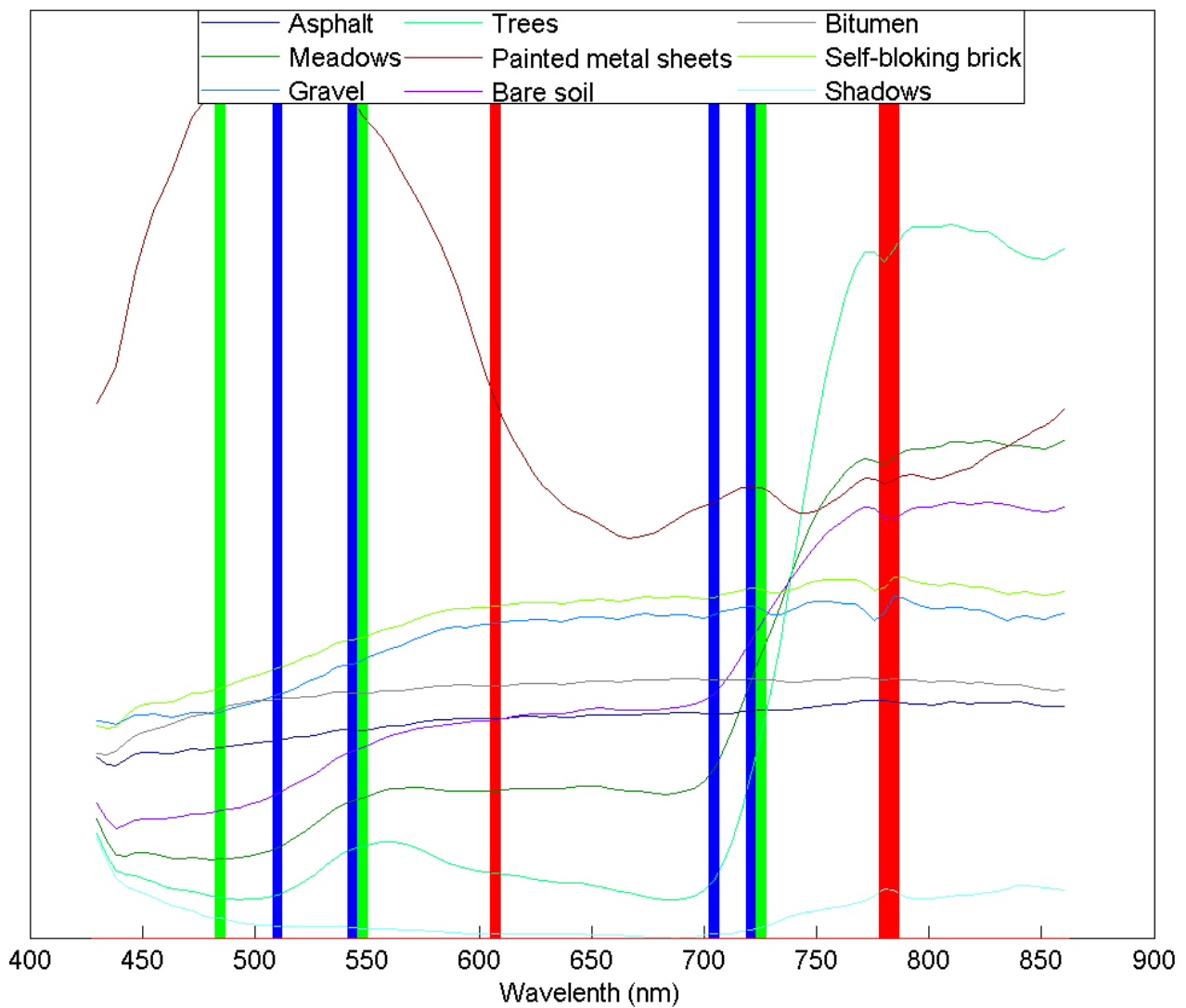


Figure 12: selected features on the University of Pavia data set ( $n_c = 200$ ) from the simple forward algorithms. Green bars correspond the feature selected by the k-CV, blue bars correspond to the feature selected by the LOOCV, red bars correspond to the feature selected by both the k-CV and the LOOCV. The mean value of each class are represented in continuous colored lines.

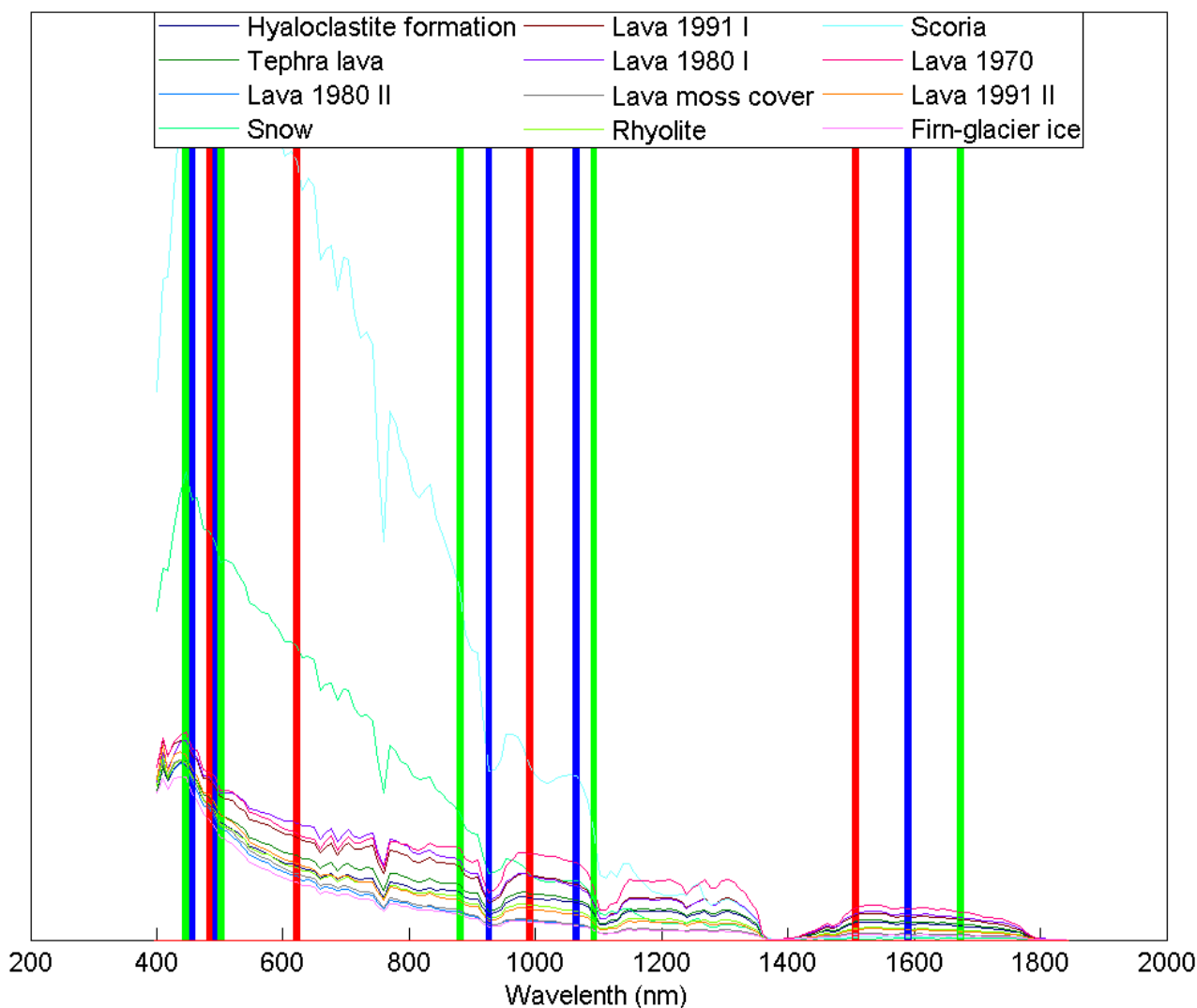


Figure 13: selected features on the Hekla data set ( $n_c = 200$ ) from the simple forward algorithms. Green bars correspond the feature selected by the k-CV, blue bars correspond to the feature selected by the LOOCV, red bars correspond to the feature selected by both the k-CV and the LOOCV. The mean value of each class are represented in continuous colored lines.

Table 7: Mean and variance of the number of selected features by both the k-CV and the LOOCV.

$n_c = 50$		
Image	Hekla	Uni
Number of features selected	$2.1 \pm 1.4$	$1.7 \pm 1.3$
$n_c = 100$		
Image	Hekla	Uni
Number of features selected	$2.8 \pm 2.1$	$2.3 \pm 1.5$
$n_c = 200$		
Image	Hekla	Uni
Number of features selected	$3.7 \pm 2.8$	$2.4 \pm 1.6$

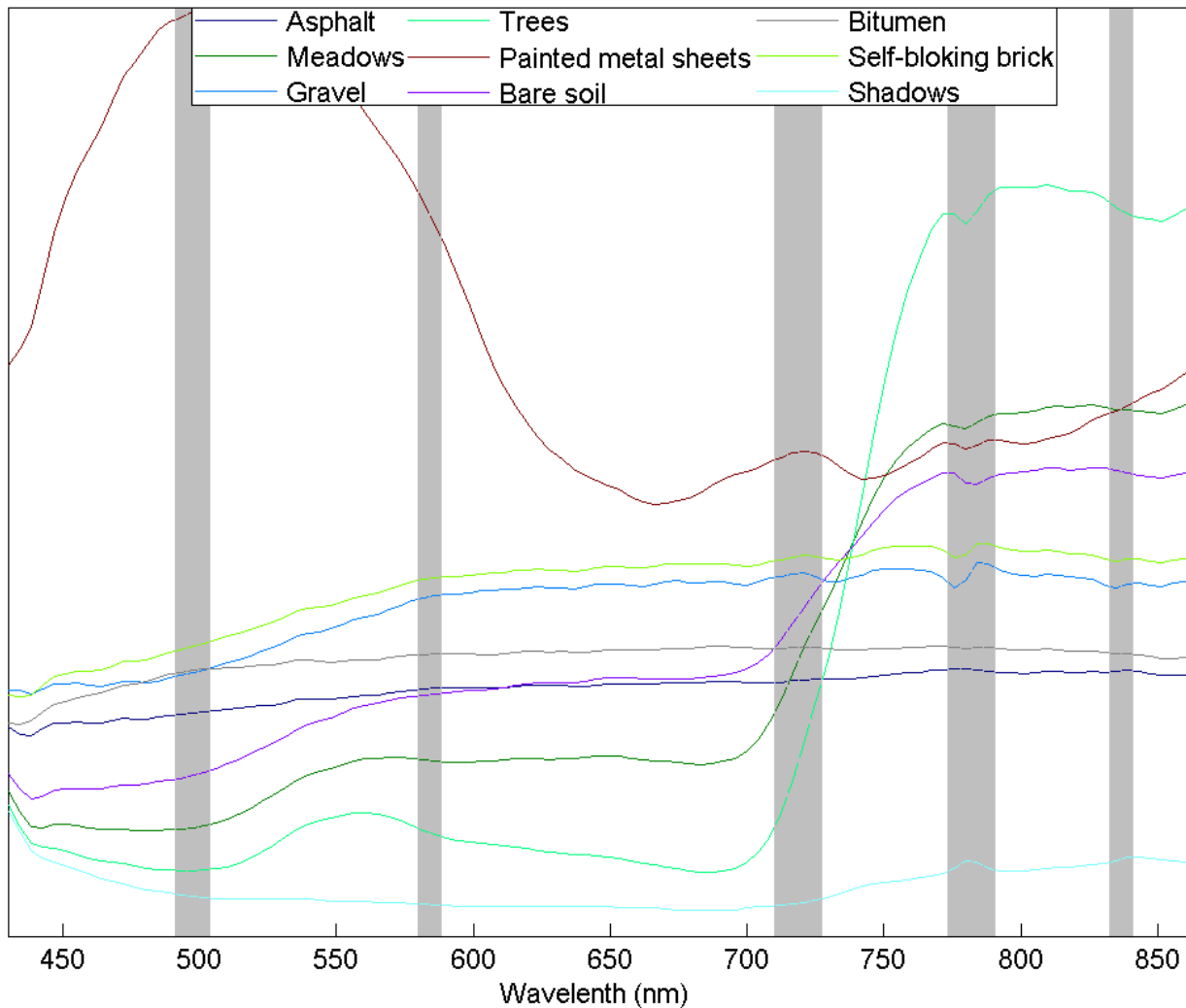


Figure 14: Most selected spectral domain for the University of Pavia. Gray bars correspond to the most selected part of the spectral domain. The mean value of each class are represented in continuous colored lines.

### 4.3 Computation time

The main difference between the two algorithms is on the computation time; the k-CV is faster than the LOOCV. This is due to the number of time the model need to be updated; for the LOOCV, the model need to be updated  $n$  times for one feature removal (where  $n$  is the number of samples) while it must be updated only  $k$  times for one feature removal for the k-CV (in general,  $k$  is a small integer). For the University of Pavia data set with  $n_c = 200$ , the k-CV takes on average 2.9 seconds to select features while the LOOCV takes on average 34.3 seconds to select features. For the Hekla data set with  $n_c = 200$ , the k-CV takes on average 8.6 seconds to select features while the LOOCV takes on average 100 seconds to select features.

## Conclusion

During this internship, traditional sparse classification methods such as SVM have been applied on hyperspectral images. Based on these methods, some algorithms have been investigated to select features, without great success. A new method of feature selection has been developed; the LOOCV NPFS. It has been tested, improved and established itself as a good sparse method to classify non linearly hyperspectral images and select features. Indeed, the linear SVM has poor results and select too many features, even when the data are projected into a polynomial space. The linear SVM has reasonable computation time when it is applied on the original data but the computation time explode when the linear SVM is applied to projected data. The nonlinear SVM has the greatest classification accuracy but does not allow to select features and has important computation time. When a RFE is applied to the nonlinear SVM, the number of remaining feature is still big and computation time are bigger. The simple forward algorithms are the most adapted to the problematic of the internship; is has a good classification rate, select few features and has small computation time. The extension of the LOOCV to k-CV also allows to treat data with a lot of samples with better accuracy. For better results, the LOOCV can be used when few samples are available and the k-CV can be used when a lot of samples are available.

Further work could be the improvement of the simple forward algorithms with a backward step to ensure that the selected features are are the most pertinent. A method to remove the variability due to the high correlation between adjacent bands. A continuous interval selection strategy, such as in [8], could be implemented. Other study could be done to take into account the spatial information. A further work can be done on the study of the extracted bands; indeed, we can see in figure 14 that the red-edge (between 700nm and 750nm) is extracted. It is a marker of plant activity [23]. It would be also interesting to apply the algorithms presented in this internship to multi-temporal data, and interpret the extracted dates.

For me, this internship has been a good experience; it has allowed me to improve my knowledge on classification methods and how to apply them on hyperspectral images. I also learned a lot about sparse models and sparse methods for feature selection. This internship strengthened my liking for the research in remote sensing and will be helpful for the doctoral thesis about the extraction of forest characteristics by joint analysis of multi-spectral or hyperspectral imaging and 3D LIDAR data I will make for the next three years.

## References

- [1] 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.
- [2] M. Fauvel, A. Zullo, and F. Ferraty, "Nonlinear parsimonious feature selection for the classification of hyperspectral images," in *Workshop on Hyperspectral image and signal processing: evolution in remote sensing*, 2014. [Online]. Available: <http://prodinra.inra.fr/record/261441>
- [3] C. J. Burges, *Dimension Reduction*. Now Publishers Inc, 2010.
- [4] L. O. Jimenez and D. A. Landgrebe, "Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 28, no. 1, pp. 39–54, feb 1998.
- [5] A. Cheriyyadat and L. M. Bruce, "Why principal component analysis is not an appropriate feature extraction method for hyperspectral data," in *Geoscience and Remote Sensing Symposium, 2003. Proceedings*, vol. 6, July 2003, pp. 3420–3422 vol.6.
- [6] A. Villa, J. A. Benediktsson, J. Chanussot, and C. Jutten, "Hyperspectral image classification with independent component discriminant analysis," vol. 49, no. 12, pp. 4865–4876, Dec 2011.
- [7] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, "Spectral and Spatial Classification of Hyperspectral Data Using SVMs and Morphological Profiles," vol. 46, no. 11 - part 2, pp. 3804–3814, Oct. 2008.
- [8] S. Serpico and G. Moser, "Extraction of spectral channels from hyperspectral images for classification purposes," vol. 45, no. 2, pp. 484–495, Feb 2007.
- [9] C.-I. Chang, *Hyperspectral Data Exploitation: Theory and Applications*. Wiley, 2007.
- [10] N. Chehata, A. L. Bris, and S. Najjar, "Contribution of band selection and fusion for hyperspectral classification," in *Proc. of the IEEE WHISPERS'14*, 2014.
- [11] D. Tuia, M. Volpi, M. Dalla Mura, A. Rakotomamonjy, and R. Flamary, "Automatic feature learning for spatio-spectral image classification with sparse SVM," vol. PP, no. 99, pp. 1–13, 2014.
- [12] D. Tuia, F. Pacifici, M. Kanevski, and W. J. Emery, "Classification of very high spatial resolution imagery using mathematical morphology and support vector machines," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 47, no. 11, pp. 3866–3879, 2009.
- [13] F. Mathieu, C. Jocelyn, B. Jón Atli *et al.*, "Kernel principal component analysis for the classification of hyperspectral remote sensing data over urban areas," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, 2009.
- [14] F. R. Bach and M. I. Jordan, "Kernel independent component analysis," *The Journal of Machine Learning Research*, vol. 3, pp. 1–48, 2003.
- [15] D. A. Landgrebe, *Signal theory methods in multispectral remote sensing*. John Wiley & Sons, 2005, vol. 29.
- [16] V. N. Vapnik and V. Vapnik, *Statistical learning theory*. Wiley New York, 1998, vol. 2.
- [17] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani, *The elements of statistical learning*. Springer, 2009, vol. 2, no. 1.
- [18] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.



- [19] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.
- [20] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.
- [21] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *Information Theory, IEEE Transactions on*, vol. 14, no. 1, pp. 55–63, 1968.
- [22] C. K. Williams and C. E. Rasmussen, "Gaussian processes for machine learning," *the MIT Press*, vol. 2, no. 3, p. 4, 2006.
- [23] S. Seager, E. L. Turner, J. Schafer, and E. B. Ford, "Vegetation's red edge: a possible spectroscopic biosignature of extraterrestrial plants," *Astrobiology*, vol. 5, no. 3, pp. 372–390, 2005.
- [24] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear svm," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 408–415.
- [25] C.-J. Lin, R. C. Weng, and S. S. Keerthi, "Trust region newton method for logistic regression," *The Journal of Machine Learning Research*, vol. 9, pp. 627–650, 2008.
- [26] M. Fauvel, "Spectral and spatial methods for the classification of urban remote sensing data," *Institut Technologique de Grenoble–Université d'Islande, Thèse de Doctorat*, 2007.
- [27] A. C. Rencher and W. F. Christensen, *Methods of multivariate analysis*. John Wiley & Sons, 2012, vol. 709.

# Appendices

## A Data resizing

During the optimization problem, a scalar product is computed. Resize the data is therefore needed to ensure that each band has the same influence. The bands with a large amplitude will have more impact on the classification than the band with a small amplitude if the data are not resized.

### A.1 No rescale

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^L x_i y_i$$

### A.2 Rescale between 0 and 1

$$\tilde{x} = \frac{x - m}{M - m}$$

$$\begin{aligned} \langle \tilde{x}, \tilde{y} \rangle &= \left\langle \frac{x - m}{M - m}, \frac{y - m}{M - m} \right\rangle \\ &= \left\langle \frac{x}{M - m}, \frac{y}{M - m} \right\rangle - \left\langle \frac{m}{M - m}, \frac{x}{M - m} \right\rangle - \left\langle \frac{m}{M - m}, \frac{y}{M - m} \right\rangle + \left\langle \frac{m}{M - m}, \frac{m}{M - m} \right\rangle \\ &= \sum_{i=1}^L \frac{1}{(M_i - m_i)^2} (x_i y_i - m_i (x_i + y_i) + m_i^2) \\ &= \underbrace{\sum_{i=1}^L \frac{x_i y_i}{(M_i - m_i)^2}}_{\langle x, y \rangle} - \underbrace{\sum_{i=1}^L \frac{m_i x_i}{(M_i - m_i)^2}}_{\langle m, x \rangle} - \underbrace{\sum_{i=1}^L \frac{m_i y_i}{(M_i - m_i)^2}}_{\langle m, y \rangle} + \underbrace{\sum_{i=1}^L \frac{m_i m_i}{(M_i - m_i)^2}}_{\|m\|^2} \end{aligned}$$

Where  $m = \min(\mathbf{x}) \in \mathbb{R}^L$  and  $M = \max(\mathbf{x}) \in \mathbb{R}^L$ .

### A.3 Rescale between -1 and 1

$$\tilde{x} = 2 \times \frac{x - m}{M - m} - 1$$

$$\begin{aligned}
 \langle \tilde{x}, \tilde{y} \rangle &= \langle 2 \times \frac{x-m}{M-m} - 1, 2 \times \frac{y-m}{M-m} - 1 \rangle = 4 \langle \frac{x-\mu}{M-m}, \frac{y-\mu}{M-m} \rangle \\
 &= \sum_{i=1}^L \frac{1}{(M_i - m_i)^2} (4x_i y_i + 2(M_i m_i - (M_i + m_i)(x_i + y_i)) + M_i^2 + m_i^2) \\
 &= 4 \sum_{i=1}^L \frac{1}{(M_i - m_i)^2} (x_i y_i - \mu_i (x_i + y_i) + \mu_i^2) \\
 &= 4 \left( \underbrace{\sum_{i=1}^L \frac{x_i y_i}{(M_i - m_i)^2}}_{\langle x, y \rangle} - \underbrace{\sum_{i=1}^L \frac{\mu_i x_i}{(M_i - m_i)^2}}_{\langle \mu, x \rangle} - \underbrace{\sum_{i=1}^L \frac{\mu_i y_i}{(M_i - m_i)^2}}_{\langle \mu, y \rangle} + \underbrace{\sum_{i=1}^L \frac{\mu_i \mu_i}{(M_i - m_i)^2}}_{\|\mu\|^2} \right) \\
 &\quad \text{with } \mu = \frac{M+m}{2}
 \end{aligned}$$

Where  $m = \min(x) \in \mathbb{R}^L$  and  $M = \max(x) \in \mathbb{R}^L$ .

#### A.4 Standardize

$$\begin{aligned}
 \tilde{x} &= \frac{x - \mu}{\sigma} \\
 \langle \tilde{x}, \tilde{y} \rangle &= \langle \frac{x - \mu}{\sigma}, \frac{y - \mu}{\sigma} \rangle \\
 &= \langle \frac{x}{\sigma}, \frac{y}{\sigma} \rangle - \langle \frac{\mu}{\sigma}, \frac{x}{\sigma} \rangle - \langle \frac{\mu}{\sigma}, \frac{y}{\sigma} \rangle + \langle \frac{\mu}{\sigma}, \frac{\mu}{\sigma} \rangle \\
 &= \sum_{i=1}^L \frac{1}{\sigma_i^2} (x_i y_i - \mu_i (x_i + y_i) + \mu_i^2) \\
 &= \underbrace{\sum_{i=1}^L \frac{x_i y_i}{\sigma_i^2}}_{\langle x, y \rangle} - \underbrace{\sum_{i=1}^L \frac{\mu_i x_i}{\sigma_i^2}}_{\langle \mu, x \rangle} - \underbrace{\sum_{i=1}^L \frac{\mu_i y_i}{\sigma_i^2}}_{\langle \mu, y \rangle} + \underbrace{\sum_{i=1}^L \frac{\mu_i \mu_i}{\sigma_i^2}}_{\|\mu\|^2}
 \end{aligned}$$

Where  $\mu \in \mathbb{R}^L$  is the mean of  $x$  and  $\sigma \in \mathbb{R}^L$  is the standard deviation of  $x$ .

#### A.5 Effect of the resizing

The drawback of using  $m = \min(x) \in \mathbb{R}^L$  and  $M = \max(x) \in \mathbb{R}^L$  is that an outlier will lead to poor results. The standardization is the resizing operation that lead to the best results (see table 8).

Table 8: Effect of the resizing; accuracy for the University data set, LIBLINEAR; L1-regularized, Logistic-loss, 50 samples per class.

Not resized	Rescaled between 0 and 1	Rescaled between -1 and 1	Standardized
73.0 ± 3.0	73.3 ± 2.7	74.0 ± 2.9	75.1 ± 2.5

On figure 15, we can observe the impact of the different resizing methods on real data. We can see big variation between the bands on the original data. The rescaling between 0 and 1 (figure 15c) or between -1 and 1 (figure 15d) reduce this variation. The standardization corrects the more effectively this problem (figure 15b); the variation between the bands is nearly suppressed.

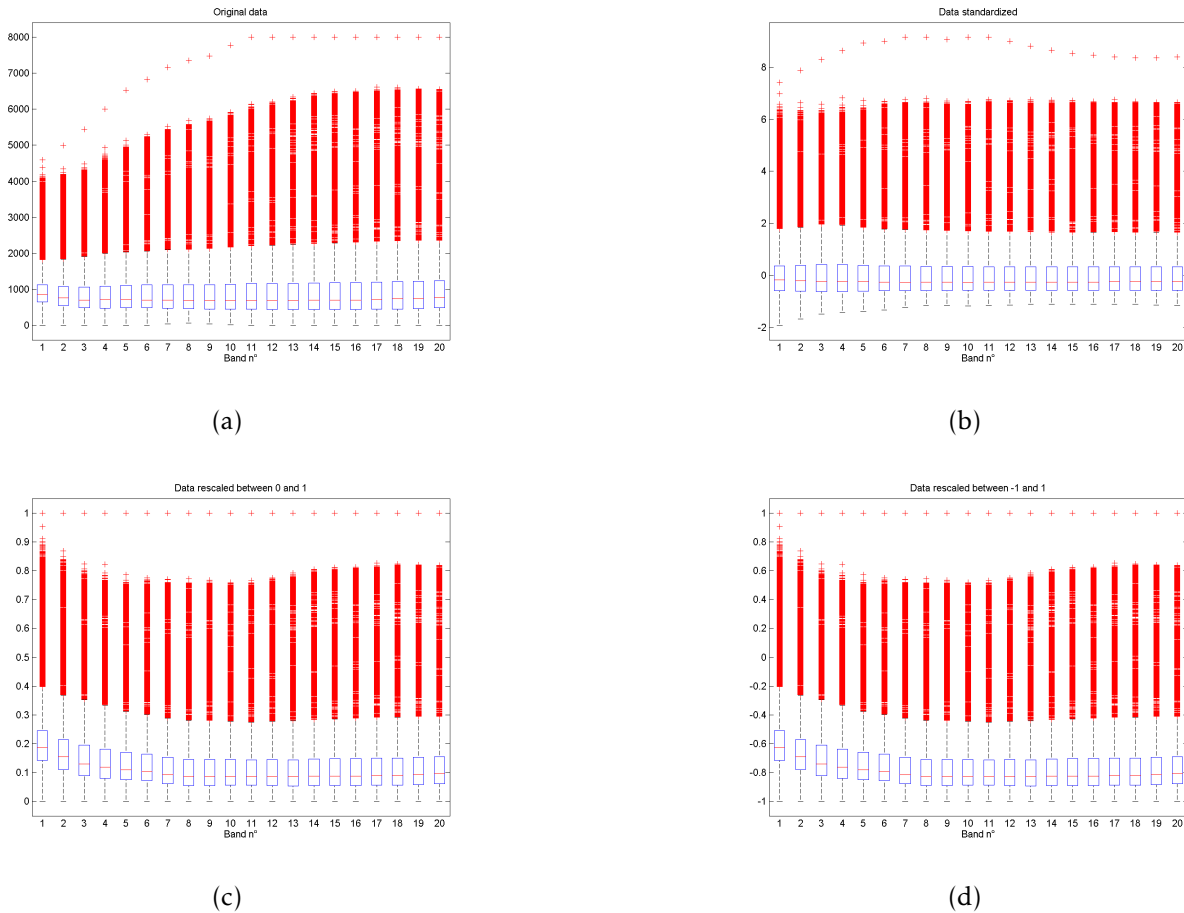


Figure 15: Effect of the resizing of the data on the 20 first bands of the University data set.

## B Simple forward feature selection

### B.1 Notations

Here, the following notations are used.  $n$  : number of samples.

$n_c$  : number of samples in class  $c$ .

$\pi_c^n$  : proportion of the samples of class  $c$  relative to the  $n$  total samples.

$\mu_c^{n_c}$  : mean vector of the class  $c$  containing  $n_c$  samples.

$\Sigma_c^{n_c}$  : covariance matrix of the class  $c$  containing  $n_c$  samples.

$\mu_c^r$  : mean vector of the  $v$  samples removed from the class  $c$ .

$\Sigma_c^r$  : covariance matrix of the  $v$  samples removed from the class  $c$ .

$$\mu_c^r = \frac{1}{v} \sum_{j=n_c-v+1}^{n_c} x_j$$

$$\Sigma_c^r = \frac{1}{v} \sum_{j=n_c-v+1}^{n_c} (\mathbf{x}_j - \boldsymbol{\mu}_c^r)(\mathbf{x}_j - \boldsymbol{\mu}_c^r)^T$$

## B.2 Estimators

### B.2.1 Proportion

$$\pi_c^n = \frac{n_c}{n} \quad (21)$$

### B.2.2 Mean vector

$$\boldsymbol{\mu}_c^{n_c} = \frac{1}{n_c} \sum_{i=1}^{n_c} \mathbf{x}_i \quad (22)$$

### B.2.3 Covariance matrix

$$\Sigma_c^{n_c} = \frac{1}{n_c} \sum_{i=1}^{n_c} (\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c})(\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c})^T \quad (23)$$

### B.2.4 Decision rule

$$Q_k(\mathbf{x}) = -(\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) - \ln(|\Sigma_k|) + 2 \ln(\pi_k) \quad (24)$$

## B.3 Updating formulas

We need to estimate the proportion, the mean vector and the covariance matrix when  $v$  samples are removed from the learning set.

### B.3.1 Proportion

If the  $v$  removed samples  $\mathbf{x}_j, j = n_c - v + 1, \dots, n_c$  belong to the class  $c$

$$\begin{aligned} \pi_c^{n-v} &= \frac{n_c - v}{n - v} \\ &= \frac{n\pi_c^n - v}{n - v} \end{aligned}$$

else

$$\begin{aligned} \pi_c^{n-v} &= \frac{n_c}{n - v} \\ &= \frac{n\pi_c^n}{n - v} \end{aligned}$$

### B.3.2 Mean estimation

If the  $v$  removed samples  $x_j, j = n_c - v + 1, \dots, n_c$  belong to the class  $c$

$$\begin{aligned}\mu_c^{n_c} &= \frac{1}{n_c} \sum_{i=1}^{n_c} x_i \\ &= \frac{1}{n_c} \sum_{i=1}^{n_c-v} x_i + \frac{1}{n_c} \sum_{j=n_c-v+1}^{n_c} x_j \\ &= \frac{n_c-v}{n_c} \mu_c^{n_c-v} + \frac{1}{n_c} \sum_{j=n_c-v+1}^{n_c} x_j\end{aligned}$$

$$\mu_c^{n_c} = \mu_c^{n_c-v} + \frac{v}{n_c} (\mu_c^r - \mu_c^{n_c-v}) \quad (25)$$

$$\mu_c^{n_c-v} = \frac{n_c \mu_c^{n_c} - v \mu_c^r}{n_c - v} \quad (26)$$

### B.3.3 Covariance matrix estimation

If the  $v$  removed samples  $\mathbf{x}_j$ ,  $j = n_c - v + 1, \dots, n_c$  belong to the class  $c$

$$\begin{aligned}
 \Sigma_c^{n_c} &= \frac{1}{n_c} \sum_{i=1}^{n_c} (\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c})(\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c})^T \\
 &= \frac{1}{n_c} \sum_{i=1}^{n_c} \left( \mathbf{x}_i - \boldsymbol{\mu}_c^{n_c-v} - \frac{1}{n_c} \underbrace{\sum_{j=n_c-v+1}^{n_c} (\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-v})}_S \right) \left( \mathbf{x}_i - \boldsymbol{\mu}_c^{n_c-v} - \frac{1}{n_c} \underbrace{\sum_{j=n_c-v+1}^{n_c} (\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-v})}_S \right)^T \\
 &= \frac{1}{n_c} \sum_{i=1}^{n_c} \left( (\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c-v})(\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c-v})^T + \frac{1}{n_c^2} S S^T - \frac{1}{n_c} (\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c-v}) S^T - \frac{1}{n_c} S (\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c-v})^T \right) \\
 &= \frac{1}{n_c} \sum_{i=1}^{n_c} \left( (\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c-v})(\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c-v})^T \right) + \frac{1}{n_c^2} S S^T - \frac{1}{n_c^2} \sum_{i=1}^{n_c} \left( (\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c-v}) S^T + S (\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c-v})^T \right) \\
 &= \frac{1}{n_c} \sum_{i=1}^{n_c-v} \left( (\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c-v})(\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c-v})^T \right) + \frac{1}{n_c} \sum_{j=n_c-v+1}^{n_c} \left( (\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-v})(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-v})^T \right) + \frac{1}{n_c^2} S S^T \\
 &\quad - \frac{1}{n_c^2} \sum_{i=1}^{n_c} \left( (\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c-v}) S^T + S (\mathbf{x}_i - \boldsymbol{\mu}_c^{n_c-v})^T \right) \\
 &= \frac{n_c-v}{n_c} \Sigma_c^{n_c-v} + \frac{1}{n_c} \sum_{j=n_c-v+1}^{n_c} \left( (\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-v})(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-v})^T \right) + \frac{1}{n_c^2} S S^T \\
 &\quad - \frac{1}{n_c} \left( (\boldsymbol{\mu}_c^{n_c} - \boldsymbol{\mu}_c^{n_c-v}) S^T + S (\boldsymbol{\mu}_c^{n_c} - \boldsymbol{\mu}_c^{n_c-v})^T \right) \\
 &= \frac{n_c-v}{n_c} \Sigma_c^{n_c-v} + \frac{1}{n_c} \sum_{j=n_c-v+1}^{n_c} \left( (\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-v})(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-v})^T \right) + \frac{1}{n_c^2} S S^T - \frac{2}{n_c^2} S S^T \\
 &= \frac{n_c-v}{n_c} \Sigma_c^{n_c-v} + \frac{1}{n_c} \sum_{j=n_c-v+1}^{n_c} \left( (\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-v})(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-v})^T \right) \\
 &\quad - \frac{1}{n_c^2} \sum_{j=n_c-v+1}^{n_c} (\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-v}) \sum_{j=n_c-v+1}^{n_c} (\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-v})^T \\
 &= \frac{n_c-v}{n_c} \Sigma_c^{n_c-v} + \frac{1}{n_c} \sum_{j=n_c-v+1}^{n_c} \left( (\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-v})(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-v})^T \right) \\
 &\quad - \frac{1}{(n_c-v)^2} \sum_{j=n_c-v+1}^{n_c} (\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c}) \sum_{j=n_c-v+1}^{n_c} (\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c})^T \\
 &= \frac{n_c-v}{n_c} \Sigma_c^{n_c-v} + \frac{v}{n_c} \Sigma_c^r + \frac{n_c v}{(n_c-v)^2} (\boldsymbol{\mu}_c^r - \boldsymbol{\mu}_c^{n_c})(\boldsymbol{\mu}_c^r - \boldsymbol{\mu}_c^{n_c})^T - \frac{v^2}{(n_c-v)^2} (\boldsymbol{\mu}_c^r - \boldsymbol{\mu}_c^{n_c})(\boldsymbol{\mu}_c^r - \boldsymbol{\mu}_c^{n_c})^T \\
 &= \frac{n_c-v}{n_c} \Sigma_c^{n_c-v} + \frac{v}{n_c} \Sigma_c^r + \frac{v}{n_c-v} (\boldsymbol{\mu}_c^r - \boldsymbol{\mu}_c^{n_c})(\boldsymbol{\mu}_c^r - \boldsymbol{\mu}_c^{n_c})^T
 \end{aligned}$$

$$\Sigma_c^{n_c-v} = \frac{n_c}{n_c-v} \Sigma_c^{n_c} - \frac{v}{n_c-v} \Sigma_c^r - \frac{n_c v}{(n_c-v)^2} (\mu_c^r - \mu_c^{n_c})(\mu_c^r - \mu_c^{n_c})^T \quad (27)$$

### Useful formulas

$$\sum_{j=n_c-v+1}^{n_c} (\mathbf{x}_j - \mu_c^{n_c}) = v(\mu_c^r - \mu_c^{n_c})$$

$$\begin{aligned} S &= n_c(\mu_c^{n_c} - \mu_c^{n_c-v}) \\ &= \sum_{j=n_c-v+1}^{n_c} (\mathbf{x}_j - \mu_c^{n_c-v}) \\ &= \sum_{j=n_c-v+1}^{n_c} \left( \mathbf{x}_j - \frac{n_c \mu_c^{n_c} - \sum_{k=n_c-v+1}^{n_c} \mathbf{x}_k}{n_c - v} \right) \\ &= \frac{1}{n_c - v} \sum_{j=n_c-v+1}^{n_c} \left( (n_c - v) \mathbf{x}_j - n_c \mu_c^{n_c} + \sum_{k=n_c-v+1}^{n_c} \mathbf{x}_k \right) \\ &= \frac{n_c}{n_c - v} \sum_{j=n_c-v+1}^{n_c} (\mathbf{x}_j - \mu_c^{n_c}) \end{aligned}$$

$$\begin{aligned} \sum_{j=n_c-v+1}^{n_c} (\mathbf{x}_j - \mu_c^{n_c-v})(\mathbf{x}_j - \mu_c^{n_c-v})^T &= \sum_{j=n_c-v+1}^{n_c} \left( (\mathbf{x}_j - \mu_c^r + \mu_c^r - \mu_c^{n_c-v})(\mathbf{x}_j - \mu_c^r + \mu_c^r - \mu_c^{n_c-v})^T \right) \\ &= v \Sigma_c^r + v(\mu_c^r - \mu_c^{n_c-v})(\mu_c^r - \mu_c^{n_c-v})^T + \underbrace{\sum_{j=n_c-v+1}^{n_c} \left( (\mathbf{x}_j - \mu_c^r)(\mu_c^r - \mu_c^{n_c-v})^T \right)}_0 \\ &\quad + \underbrace{\sum_{j=n_c-v+1}^{n_c} \left( (\mu_c^r - \mu_c^{n_c-v})(\mathbf{x}_j - \mu_c^r)^T \right)}_0 \\ &= v \Sigma_c^r + v \left( \mu_c^r - \frac{n_c \mu_c^{n_c} - v \mu_c^r}{n_c - v} \right) \left( \mu_c^r - \frac{n_c \mu_c^{n_c} - v \mu_c^r}{n_c - v} \right)^T \\ &= v \Sigma_c^r + \frac{n_c^2 v}{(n_c - v)^2} (\mu_c^r - \mu_c^{n_c})(\mu_c^r - \mu_c^{n_c})^T \end{aligned}$$



## B.4 LOOCV algorithm

---

**Algorithm 3** Simple forward LOOCV feature selection.

---

**Input:**  $S, \delta, \text{maxvariables}$

**Output:** The set of selected variables  $id$

$C \leftarrow$  number of classes

$n \leftarrow$  number of samples

$d \leftarrow$  number of variables

$rep \leftarrow 1$

$id \leftarrow []$  // Pool of selected features

$variable \leftarrow [1, \dots, d]$  // Original features

Compute model for each class using eq. (12), (13) and (14)

**while**  $rep \leq \text{maxvariable}$  **do**

$nv = \text{length}(variable)$  // Number of remaining variables

$loocv \leftarrow [0, \dots, 0]$  // Vector of size  $nv$

**for**  $j = 1, \dots, nv$  **do**

$id\_t \leftarrow [id, variable(j)]$

        Compute the decision function with the marginalized model

$loocv\_tp \leftarrow 0$

**for**  $i = 1, \dots, n$  **do**

**for**  $k = 1, \dots, C$  **do**

**if**  $y_i = k$  **then**

                    Update the model

                    Compute the decision function

**else**

                    Compute the decision function

**end if**

**end for**

$yl_{oo} \leftarrow \arg \max_{k=1, \dots, C} Q_k(x_i)$

$loocv\_tp \leftarrow loocv\_tp + (yl_{oo} = y_i)$

**end for**

$loocv(j) \leftarrow \frac{loocv\_tp}{n}$

**end for**

    Get the maximum of  $loocv$  and the corresponding variable  $t$

**if** The improvement in terms of  $loocv < \delta$  **then**

**break**

**else**

        Add the variable  $t$  to the pool  $id$

        Remove the variable  $t$  from the original set of variables

**end if**

**end while**

---

## C Figures

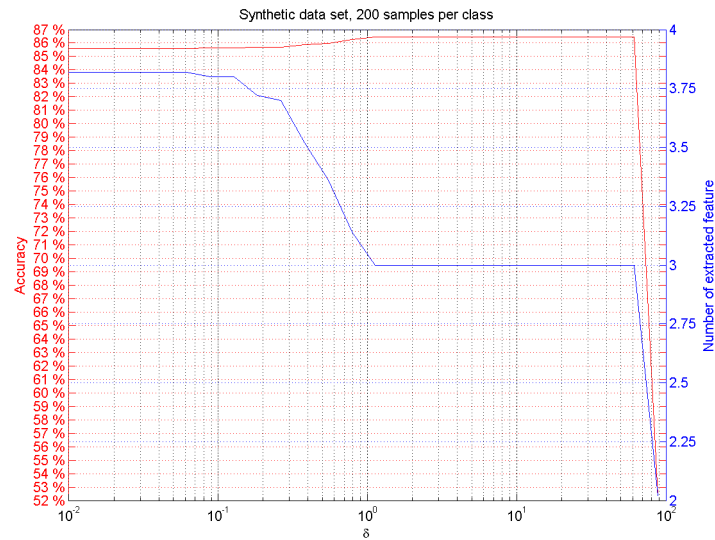


Figure 16: Accuracy and number of selected features as a function of  $\delta$  for a synthetic data set (3 discriminant variables over the 200 available, 8 classes) using the leave-one-out method.

The greatest accuracy is obtained when the 3 discriminant variables are selected, otherwise, the accuracy decrease.

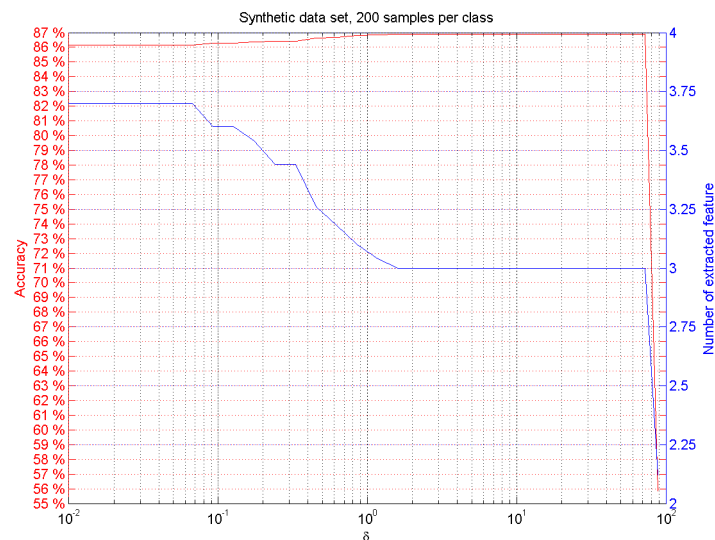


Figure 17: Accuracy and number of selected features as a function of  $\delta$  for a synthetic data set (3 discriminant variables over the 200 available, 8 classes) using the k-fold method.

The greatest accuracy is obtained when the 3 discriminant variables are selected, otherwise, the accuracy decrease.

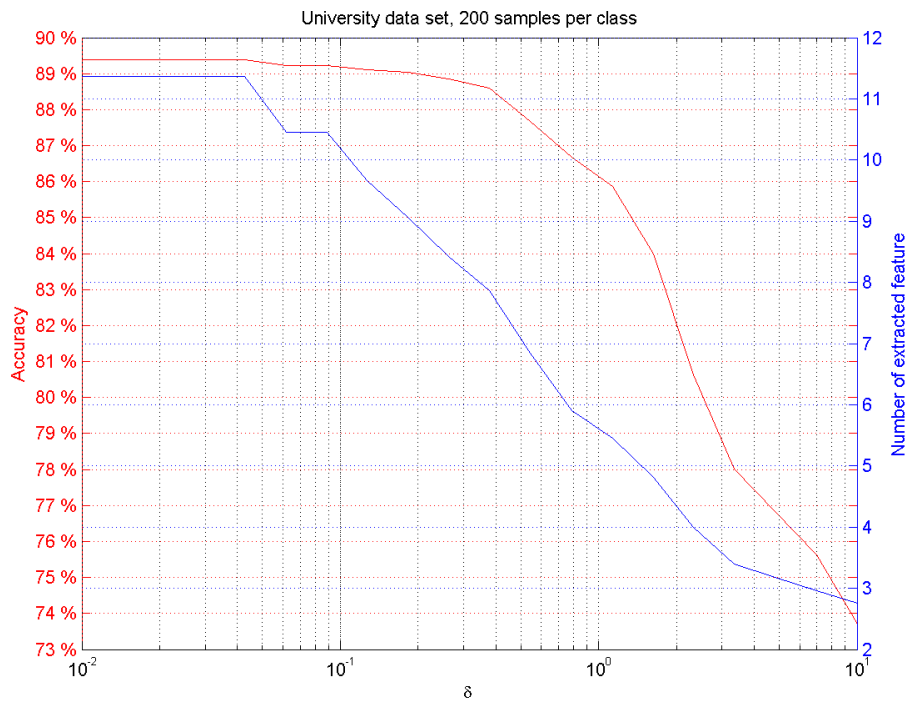


Figure 18: Accuracy and number of selected features as a function of  $\delta$  for the University data set using the leave-one-out method.

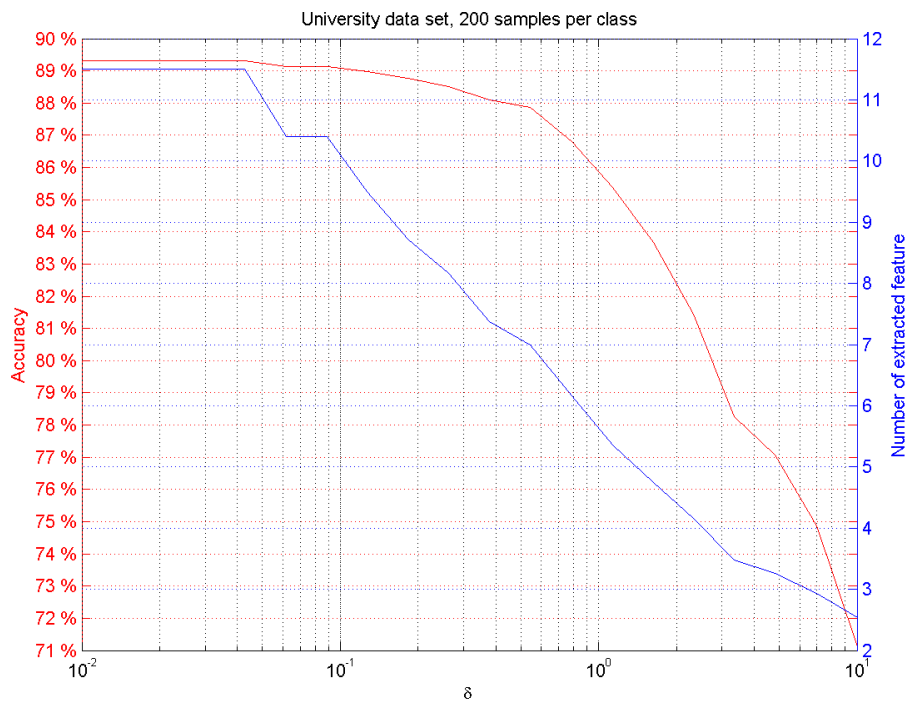


Figure 19: Accuracy and number of selected features as a function of  $\delta$  for the University data set using the k-fold method.

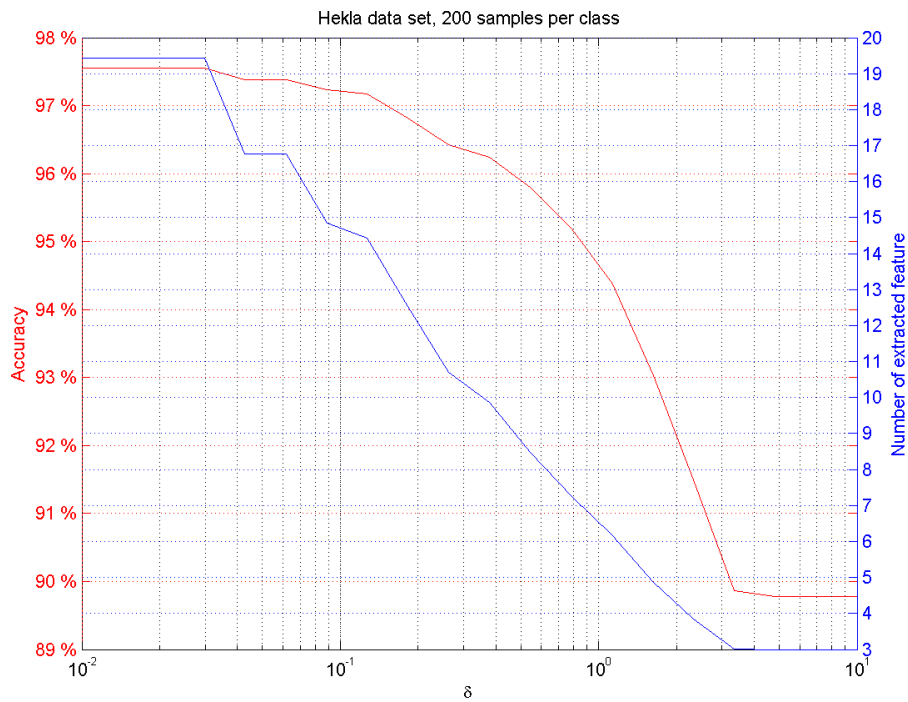


Figure 20: Accuracy and number of selected features as a function of  $\delta$  for the Hekla data set using the leave-one-out method.

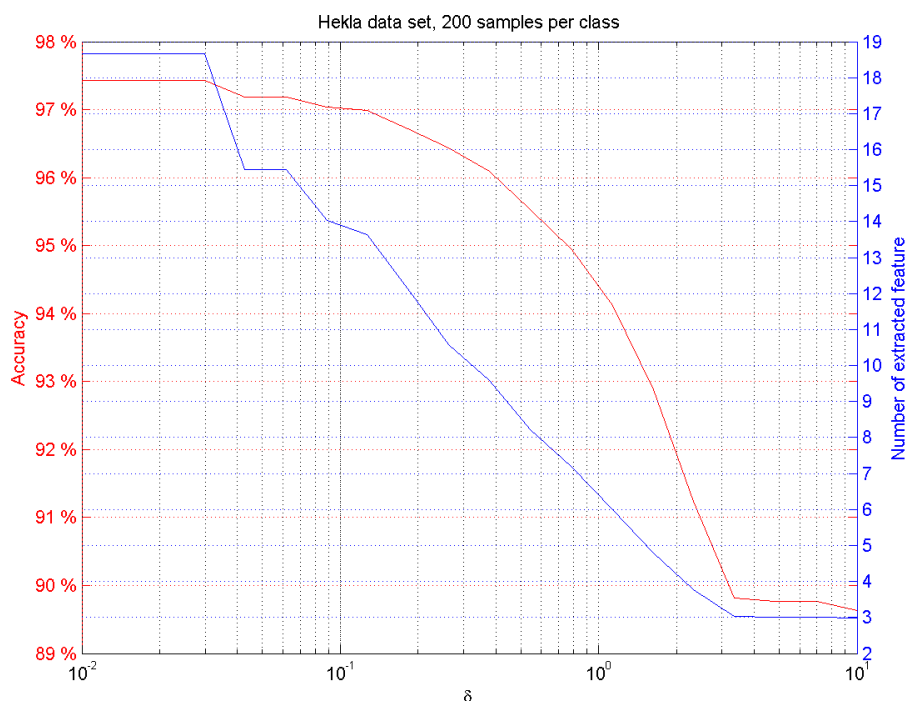


Figure 21: Accuracy and number of selected features as a function of  $\delta$  for the Hekla data set using the k-fold method.

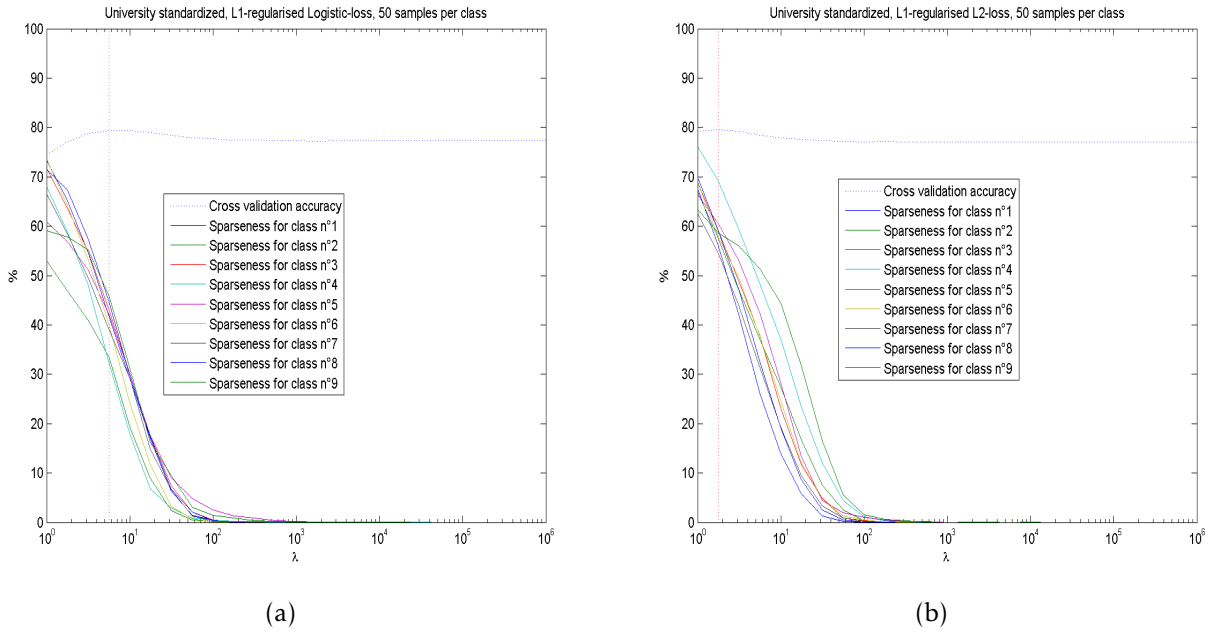


Figure 22: Sparseness of the vector  $w$  (% of zeros) for each class and cross-validation accuracy (% of good classification) as a function of the regularization parameter for the University data set using the L1-regularized linear SVM ( $n_c = 50$ ). The red dotted line correspond to the value of  $\lambda$  that corresponds to the greatest cross validation accuracy.

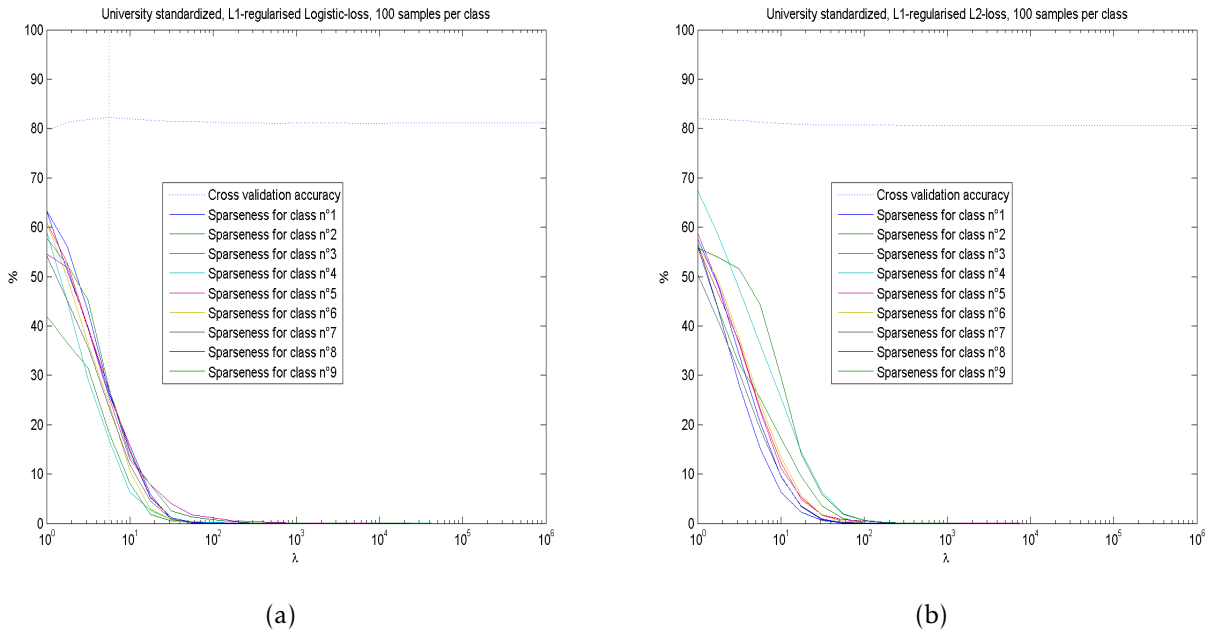


Figure 23: Sparseness of the vector  $w$  (% of zeros) for each class and cross-validation accuracy (% of good classification) as a function of the regularization parameter for the University data set using the L1-regularized linear SVM ( $n_c = 100$ ). The red dotted line correspond to the value of  $\lambda$  that corresponds to the greatest cross validation accuracy.

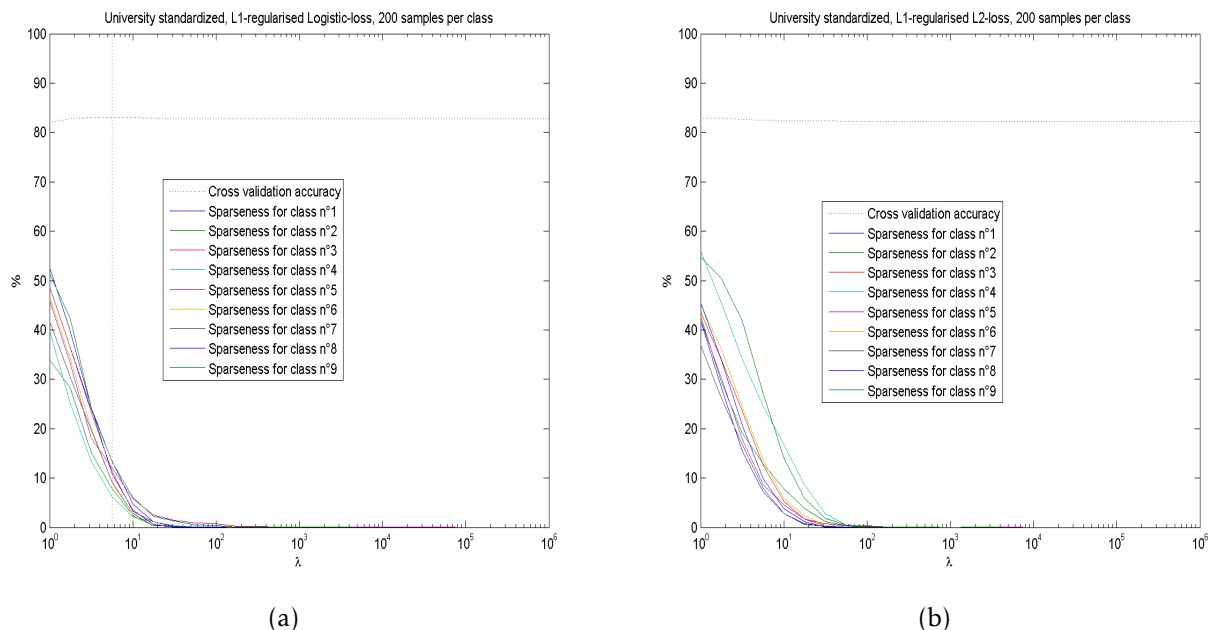


Figure 24: Sparseness of the vector  $w$  (% of zeros) for each class and cross-validation accuracy (% of good classification) as a function of the regularization parameter for the University data set using the L1-regularized linear SVM ( $n_c = 200$ ). The red dotted line correspond to the value of  $\lambda$  that corresponds to the greatest cross validation accuracy.

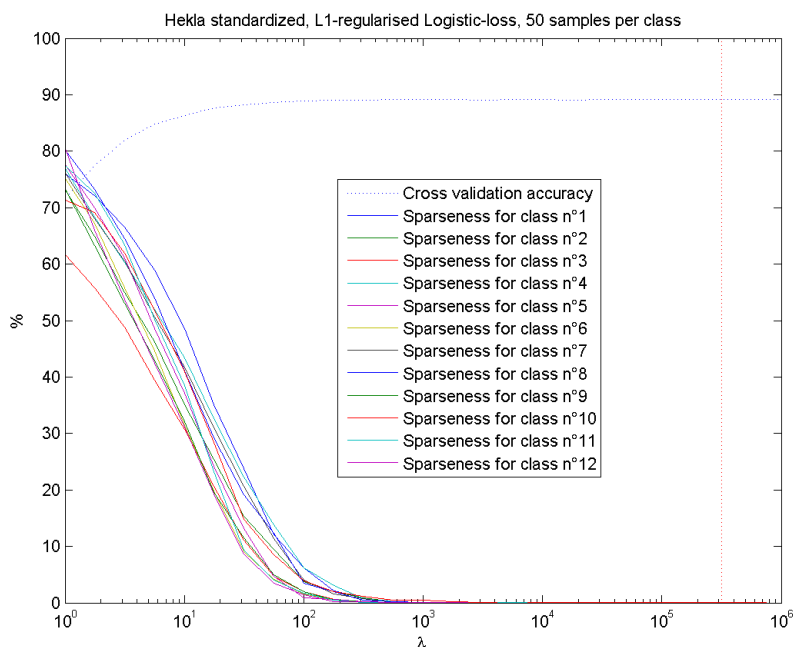


Figure 25: Sparseness of the vector  $w$  (% of zeros) for each class and cross-validation accuracy (% of good classification) as a function of the regularization parameter for the Hekla data set using the L1-regularized linear SVM ( $n_c = 50$ ). The red dotted line correspond to the value of  $\lambda$  that corresponds to the greatest cross validation accuracy.

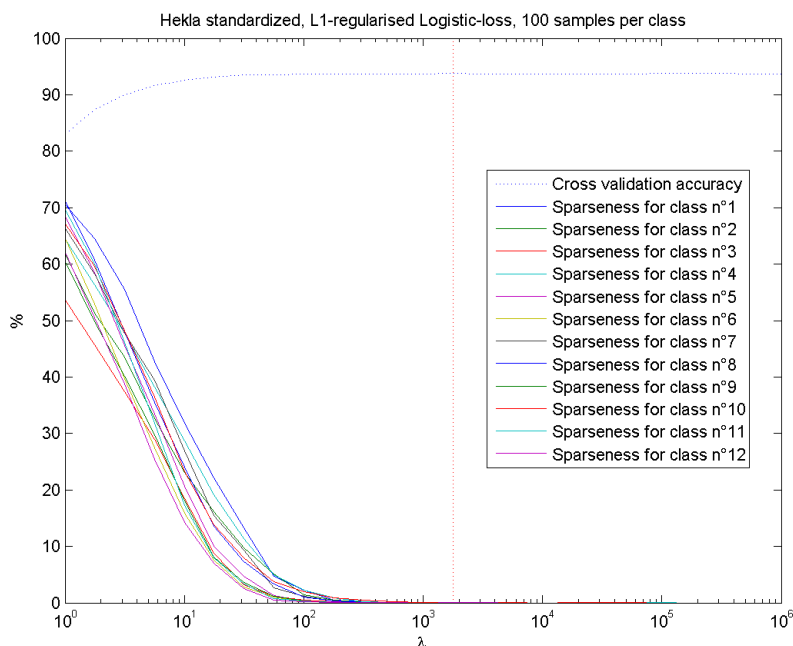


Figure 26: Sparseness of the vector  $w$  (% of zeros) for each class and cross-validation accuracy (% of good classification) as a function of the regularization parameter for the Hekla data set using the L1-regularized linear SVM ( $n_c = 100$ ). The red dotted line correspond to the value of  $\lambda$  that corresponds to the greatest cross validation accuracy.

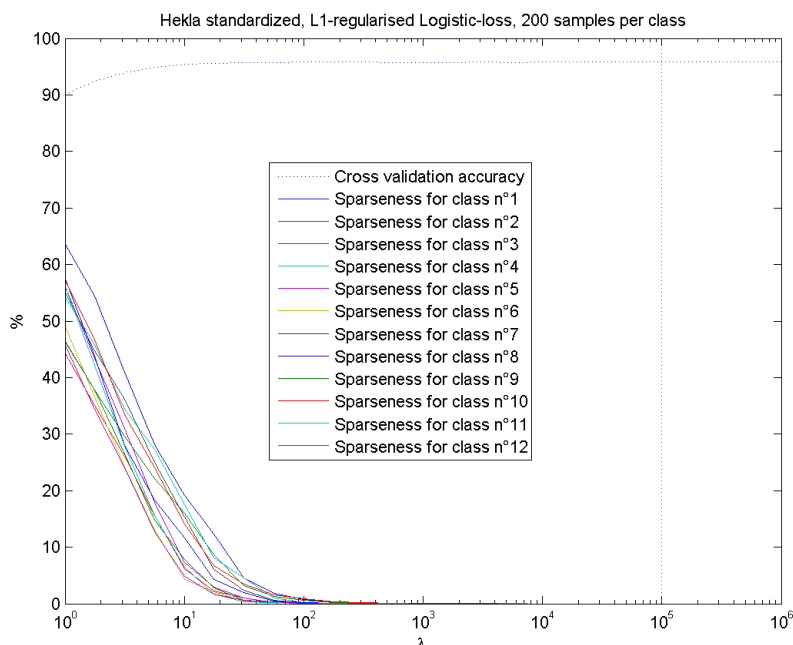


Figure 27: Sparseness of the vector  $w$  (% of zeros) for each class and cross-validation accuracy (% of good classification) as a function of the regularization parameter for the Hekla data set using the L1-regularized linear SVM ( $n_c = 200$ ). The red dotted line correspond to the value of  $\lambda$  that corresponds to the greatest cross validation accuracy.