Joint Data Filtering and Labeling using Gaussian Processes and Alternating Direction Method of Multipliers

Pablo Ruiz, *Student Member, IEEE,* Rafael Molina, *Senior Member, IEEE,* and Aggelos K. Katsaggelos, *Fellow, IEEE*

***Abstract*—Sequence labeling aims at assigning a label to every sample of a signal (or pixel of an image) while taking into account the sequentiality (or vicinity) of the samples. To perform this task, many works in the literature first filter and then label the data. Unfortunately, the filtering, which is performed independently from the labeling, is far from optimal and frequently makes the latter task harder. In this work, a novel approach which trains a Gaussian Process (GP) classifier and estimates the coefficients of an optimal filter jointly is presented. The new approach, based on Bayesian modeling and Alternating Direction Method of Multipliers (ADMM) optimization, performs both tasks simultaneously. All unknowns are treated as stochastic variables which are estimated using Variational Inference and filtering and labeling are linked with the use of ADMM. In the experimental section synthetic and real experiments are presented to compare the proposed method with other existing approaches.**

***Index Terms*—Classification, Filtering, Gaussian Processes, Bayesian Modeling, Variational Inference, ADMM.**

I. Introduction

THE accelerated acquisition of data has by far exceeded the human capacity to transform them into information. This problem is shared by most problems of 21st century science. The concept of Big Data [1] includes acquisition, storage, search, sharing, analysis and visualization of the data. The work of the data scientist focuses primarily on analysis and visualization, for which it is necessary to develop efficient algorithms to extract useful information. Machine Learning [2], [3] provides powerful tools to solve various problems such as regression, classification, clustering, and different perspectives to address these problems.

Many classification tasks are oriented towards performing sequence labeling, that is, assigning a label to every sample of a signal (or pixel of an image) while taking into account the sequentiality (or vicinity) of the samples. This task is normally approached by first filtering the data and then performing classification. For instance, a super resolution method can be applied to a multispectral image [4] followed by the application of a classification method on the improved multispectral image [5]; or an improved passive millimeter-wave image can be obtained [6] followed by an object detection procedure [7].

The use of spatially localized oscillating functions as linear feature extractors is ubiquitous across systems dealing with images, with a wide range of applications in computer vision and image processing. These filters, see [8] for their motivation in the visual cortex context, are in the form of filter banks, dictionaries or visual vocabularies, and they emerge when different learning machines are applied to natural images, see for instance [9]. While different criteria have been proposed to obtain such functions their remarkable success may be related to their intrinsic relevance to a fundamental problem of object classification in vision.

A variety of methods in vision employ filterbanks, where the main goal in general is the accurate representation of data instead of classification. For instance, unsupervised learning leads to this kind of filterbanks, but these approaches are not related to classification. Bag of words approaches in computer vision use this kind of filterbanks as a first stage in classification (e.g., SIFT-like features [10]), but these are heuristically defined, fixed, and not optimized together with the classifier. Enforcing sparsity via these filterbanks leads to powerful priors on images, which in turn provide good results in common image processing tasks such as denoising, inpainting, etc. However, when aiming at optimal classification, it is not clear if sparsity is a necessary condition or how it should be used [11], [12].

The recognition of real world scenes in images is a challenging problem with important consequences in content based image retrieval, object detection, and image understanding, among others. Scene recognition entails tagging the scene with a finite set of labels which provides information about its semantic category. In [13] a holistic descriptor, GIST, was introduced to label natural images in several categories. GIST is an attribute vector that represents different global properties of the image based on low-level features computed using a frequency representation of the image. Each descriptor value is computed using a linear regression model on the feature values.

Formally, we assume that we have access to a multichannel sequential signal or multichannel sequential features extracted from the signal. We use the term "multichannel features" to refer to both concepts for simplicity. Let $\mathbf{Z}_{\text{ori}} = [\mathbf{z}_1, \ldots, \mathbf{z}_P]$ be the matrix including the P input features, where each feature $\mathbf{z}_j$ is of length B. Instead of performing the classification

P. Ruiz and R. Molina, are with the Departamento de Ciencias de la Computación e I. A. E.T.S. Ing. Informática y Telecomunicación. Universidad de Granada, 18071 Granada, Spain. (e-mail: {mataran,rms}@decsai.ugr.es).

A. K. Katsaggelos is with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208 USA (e-mail: aggk@eecs.northwestern.edu).

This paper has been partially supported by the Spanish Ministry of Economy and Competitiveness under project TIN2013-43880-R, and the Department of Energy (DE-NA0002520).

task directly on the features $\mathbf{Z}_{\text{ori}}$, our goal is to compute new features $\mathbf{X}_{\text{ori}} = [\mathbf{x}_1, \ldots, \mathbf{x}_P]$ so as to optimize the classification performance. $\mathbf{Z}_{\text{ori}}$ and $\mathbf{X}_{\text{ori}}$ can be related in two different ways. The first method, referred to as **analysis representation**, obtains $\mathbf{X}_{\text{ori}}$ as a linear transformation of $\mathbf{Z}_{\text{ori}}$, leading to

$$\mathbf{x} = \mathbf{H}\mathbf{z}, \tag{1}$$

where $\mathbf{x}$ and $\mathbf{z}$ are column vectors obtained by lexicographically ordering $\mathbf{X}_{\text{ori}}$ and $\mathbf{Z}_{\text{ori}}$, respectively. In this case, $\mathbf{H}$ has the interpretation of a linear filterbank which needs to be estimated.

A second method, referred to as **synthesis representation**, aims at representing $\mathbf{Z}_{\text{ori}}$ by the use of a dictionary that has to be learnt from a set of samples, that is,

$$\mathbf{Z}_{\text{ori}} = \mathbf{D}\mathbf{X}_{\text{ori}}. \tag{2}$$

The learnt dictionary $\mathbf{D}$ is then used on all the samples to obtain the new representation. The synthesis representation problem is related, for instance, to the use of discriminative Gaussian Process Latent Variable Models (GPLVM) [14], where a linear discriminant prior on the latent variables is introduced and bears some connections with learning discriminative dictionaries (see, for instance, [15], [16]). Here, we will not explore this approach and only address the analysis representation.

The analysis representation problem appears in many signal reconstruction problems. For instance, it can be used to improve the classification of ElectroEncephaloGraphic (EEG) data in Brain-Computer Interfaces (BCI) [17] and to discover causality interaction in functional Magnetic Resonance Imaging (fMRI) [18]. In remote sensing, sensors mounted on satellites or airborne platforms acquire information on the energy reflected by the Earth. The acquired images are generally multispectral/hyperspectral, and are very valuable in terrain analysis but lead to challenging classification problems. The analysis representation is also present in problems where feature vectors have a sequential structure or consist of features with sequential structure. For instance in image segmentation or texture classification an image is locally preprocessed assigning to each pixel a vector with the output of different local and global filters [19]. This is the case when histogram-based features are used in problems like human detection where an image is divided into small spatial regions (cells) and a local 1-D histogram of gradient directions or edge orientations over the cells is calculated, leading to the Histogram of Oriented Gradient (HOG) descriptor [20]. Another example is the Bag of Word (BoW) features in image classification or categorization [21].

The idea of jointly optimizing a filter and a classifier dates back to the 1990s within the field of artificial neural networks. It was, for instance, used in convolutional networks [22] or to define a neural model for temporal processing [23]. Recently, Flamary *et al.* [24] presented a system where the filters are learnt jointly with a Support Vector Machine (SVM) to perform classification. Following the same line, we proposed a Bayesian framework to estimate the filterbank jointly with a Gaussian Process (GP) classifier [25].

In this paper we use a different model from the one presented in [25] to combine filter estimation with GP classification in a Bayesian framework as explained next. Variational inference is used to simultaneously learn the classifier, the optimal filterbank coefficients, and the model parameters.

The key step linking the filter estimation with the GP classifier is to enforce the solution to satisfy the constraint in Eq. (1). To do that in [25], we utilized a parameter $\nu \in \mathbb{R}^+$, which was increasing at each iteration of the algorithm. Meanwhile the model we are proposing here, is capable of enforce this constraint automatically. To achieve this we introduce a new variable in the Bayesian framework, which follows a probability distribution based on the Alternating Directional Method of Multipliers (ADMM) [26]. ADMM is often utilized to transform a constrained optimization problem into an unconstrained one through the use of the augmented Lagrangian. The GP Classifier prior has also been modified. In [25] we introduced a parameter in order to guarantee the kernel matrix being non-singular. In this work we have followed the solution proposed by Rasmussen et al. [27], which in our opinion, is more appropriate for applying the model to real situations. The main advantage in this work is that, unlike either [24] or [25] where the parameters must be set in advance, our new model leads to a parameter-free algorithm.

Finally, in the experimental section we apply the proposed method to pixel-labeling in remote sensing images, where, given a small set of training samples, the goal is to label the remaining pixels automatically. There are many approaches to this problem (see [28] and [29] for a complete review). Very few however tackle the filtering/classification problem, with the exception of [24] and [30].

The rest of this paper is organized as follows. In Section II Bayesian modeling is presented for the analysis representation on images. In Section III, once our modeling has been introduced, we relate it to other approaches for solving the same problem. In Section IV Variational Inference is performed and the proposed algorithm is presented. Details on the initialization of the algorithm are provided in Section V. The classification rule is introduced in Section VI. In Section VII experimental results are presented and finally Section VIII concludes the paper.

II. Bayesian Modeling

Let us assume that during the training phase of a labeling problem we have access to $N < P$ training vector features $\mathbf{z}_1, \ldots, \mathbf{z}_N$ and their corresponding labels $\mathbf{y} = [y_1, \ldots, y_N]^T$, with $y_i \in \{0, 1\}$. We assume here that each $\mathbf{z}_i$ is of length B. That is, we have access to N, B-band observations. We also assume that we have access to a $k \times k$ window around each $\mathbf{z}_i$. Notice that although we are formulating here a spatial domain problem, it can be easily extended to other dimensions.

In order to improve labeling performance, the observed features are transformed into new ones by filtering each band with a spatial filter $\mathbf{a}_i \in \mathbb{R}^{k^2}$, $i = 1, \ldots, B$, which has to be estimated. In other words, we have

$$\mathbf{X} = \mathbf{A}\mathbf{Z} = \begin{bmatrix} \mathbf{a}_1^T & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{a}_2^T & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{a}_B^T \end{bmatrix} \begin{bmatrix} \mathbf{z}_{1,1} & \mathbf{z}_{1,2} & \dots & \mathbf{z}_{1,N} \\ \mathbf{z}_{2,1} & \mathbf{z}_{2,2} & \dots & \mathbf{z}_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{z}_{B,1} & \mathbf{z}_{B,2} & \dots & \mathbf{z}_{B,N} \end{bmatrix}, \tag{3}$$

where each $\mathbf{z}_{i,j}$ is a column vector of size k^2 containing the neighborhood of the j-th sample in the i-th band. We also define $\mathbf{Z}_i = [\mathbf{z}_{i,1}, \dots, \mathbf{z}_{i,N}], i = 1, \dots, B$ which will be used in our derivations. To reduce the number of coefficients to be estimated in $\mathbf{A}$, we only perform intraband filtering. Interband filtering is not performed because the classifier utilizes multiband information.

Our filtering/labeling task becomes then equivalent to estimating $\mathbf{A}$ and classifying the data using the new features $\mathbf{X}$.

To model the classification function relating each sample $\mathbf{x}_j$ to its corresponding label y_j we follow a two stage procedure. Firstly, we introduce a set of latent variables $\mathbf{f} = [f_1, \dots, f_N]$ which are related to each y_j by a sigmoid function $\boldsymbol{\sigma}(f_j) = 1/(1 + e^{-f_j})$ and write the probability model

$$\mathrm{p}(\mathbf{y}|\mathbf{f}) = \prod_{j=1}^{N} \left(\frac{1}{1 + e^{-f_j}}\right)^{y_j} \left(\frac{e^{-f_j}}{1 + e^{-f_j}}\right)^{1-y_j}. \tag{4}$$

We are using here the sigmoid function to transform f_j to a value in the interval [0,1]. Notice that other functions like the probit function could also be used, see [27] and [31] in the GP context.

Secondly, we relate $\mathbf{f}$ to $\mathbf{X}$ using the following Gaussian process prior for $\mathbf{f}$

$$\mathrm{p}(\mathbf{f}|\mathbf{X}, \gamma) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \gamma^{-1}\mathbf{K}), \tag{5}$$

where γ is a precision parameter to be estimated and $\mathbf{K} \in \mathbb{R}^{N \times N}$ is a positive semidefinite kernel matrix whose entries are calculated using a kernel function $\mathbf{K}_{ij} = \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$. In this work we explore Linear and Gaussian kernels (see [31] for more details) which are the most commonly utilized kernels in the GP literature.

Not much prior information is available for modeling the filterbank. For this reason, in [24] only a quadratic regularization term is added to the objective function, in order to prevent overfitting. Following this approach and using the Bayesian framework we write

$$\mathrm{p}(\mathbf{A}|\boldsymbol{\alpha}) = \prod_{i=1}^{B} \mathrm{p}(\mathbf{a}_i|\alpha_i) = \prod_{i=1}^{B} \mathcal{N}(\mathbf{a}_i|\mathbf{0}, \alpha_i^{-1}\mathbf{I}_{k^2}), \tag{6}$$

where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_B)^T$ are precision coefficients to be estimated.

So far we have modeled the GP based classification and filter estimation independently; however, we link both problems by solving the constrained problem

$$\begin{aligned} &(\hat{\mathbf{f}}, \hat{\gamma}, \hat{\mathbf{A}}, \hat{\boldsymbol{\alpha}}) = \\ &\quad \mathrm{argmax}_{\mathbf{f},\gamma,\mathbf{A},\boldsymbol{\alpha}} \mathrm{p}(\mathbf{y}|\mathbf{f})\mathrm{p}(\mathbf{f}|\mathbf{X}, \gamma)\mathrm{p}(\mathbf{A}|\boldsymbol{\alpha})\mathrm{p}(\boldsymbol{\alpha})\mathrm{p}(\gamma), \\ &\quad \text{subject to} \quad \mathbf{X} = \mathbf{A}\mathbf{Z}. \end{aligned} \tag{7}$$

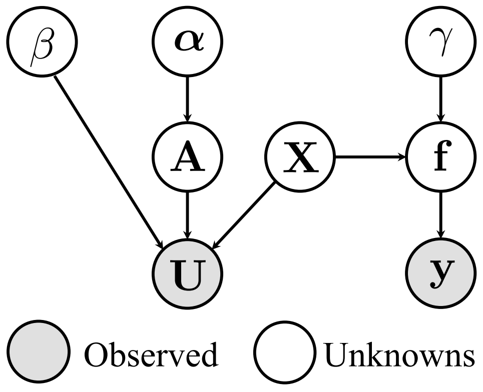


Fig. 1. Graphical model for the filtering/labeling problem, in which we assume that $\mathbf{U}$ and $\mathbf{y}$ are observed

To transform the constrained problem in Eq. (7) into an unconstrained one, we use the generalization of ADMM proposed in [32]. Consider the following pseudo-observation model

$$\mathrm{p}(\mathbf{U}|\mathbf{A}, \mathbf{X}, \beta) \propto \beta^{\frac{NB}{2}} \exp\left(-\frac{\beta}{2}\|\mathbf{U} - (\mathbf{A}\mathbf{Z} - \mathbf{X})\|_{\mathrm{F}}^2\right), \tag{8}$$

where β is a precision parameter to be estimated, $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm and $\mathbf{U}$ is the matrix containing the scaled Lagrange multipliers associated with the equality constraint of the optimization problem in Eq. (7). Notice that since $\mathbf{Z}$ is fixed, we have removed it from $\mathrm{p}(\mathbf{U}|\mathbf{A}, \mathbf{X}, \beta)$. The energy of this distribution is obtained from the objective function in [26] (see Section IV-C for an extended justification).

In Fig. 1 we assume that $\mathbf{U}$, the scaled Lagrange multipliers, together with $\mathbf{y}$ are observed and we show the graphical model associated with our filtering/labeling problem.

Using a flat, improper prior on $\mathbf{X}$ as well as on the model parameters $\boldsymbol{\alpha}$, β and γ, the joint distribution can then be written as

$$\begin{aligned} \mathrm{p}(\mathbf{y}, \mathbf{U}, \Theta) = {} & \mathrm{p}(\boldsymbol{\alpha})\mathrm{p}(\beta)\mathrm{p}(\gamma) \\ & \times \mathrm{p}(\mathbf{A}|\boldsymbol{\alpha})\mathrm{p}(\mathbf{X})\mathrm{p}(\mathbf{U}|\mathbf{A}, \mathbf{X}, \beta)\mathrm{p}(\mathbf{f}|\mathbf{X}, \gamma)\mathrm{p}(\mathbf{y}|\mathbf{f}), \end{aligned} \tag{9}$$

where $\Theta = \{\mathbf{f}, \{\mathbf{a}_i\}_{i=1}^B, \mathbf{X}, \boldsymbol{\alpha}, \beta, \gamma\}$.

III. Related Models

In this section we provide additional insights into the proposed model and relate it to other filtering/classification models in the literature.

In Fig. 2 we show how our method works. For the jth pixel, the inputs are the neighborhood vectors $\mathbf{z}_{i,j}$ with $i = 1, \dots, B$. The new feature x_{ij} is then calculated as a linear combination of the values in $\mathbf{z}_{i,j}$ using the weights defined by the filter $\mathbf{a}_i$, whose components are denoted by a_i^l, with $l = 1, \dots, k^2$. Finally, a GP classifier uses the new features to assign a label to the jth pixel.

The question we address in this paper is how the filters should be chosen so as to improve the classifier performance. The first option is to consider no-filtering, that is, let $k = 1$ and $a_i^1 = 1$, $i = 1, \dots, B$. In this case, the input features to the classifier are the jth pixel intensity values (or in general the original features unaltered). This is the standard GP classification which we denote by GPC.

Another option is to use heuristics. A filter could be chosen (i.e., select the values a_i^l) based on visual properties of the image. That is, filtered images are sought after where the class of interest can be easily recognized visually. For instance, Gómez-Chova *et al.* [30] use a 9×9 filter to eliminate multiplicative noise in a multispectral image. After the input image has been filtered, the new image is used to train a standard GP classifier.

The problem of using non-tailored filtering as a preprocessing step is that the filters are not necessarily optimal with respect to classification, that is, it is not guaranteed that the best classification performance will be attained. As we will see in Section IV, our method merges both, filter estimation and GP training, in one inference procedure. This procedure does not only ensure an increase or the "highest" classification performance, but also provides us with additional information on the problem conveyed by the estimated filter coefficients.

To link filter estimation and GP training, we use ADMM [26]. As we will see in Section IV-C, ADMM introduces the scaled Lagrange multiplier term in the objective function, to transform a constrained problem into an unconstrained one. From a Bayesian perspective, this is equivalent to considering the pseudo-observation model in Eq. (8) [32]. In [25] we proposed a different way to link filter estimation and GP classification. Although the two formulations look similar, they are conceptually very different. While here we solve a constrained problem, in [25] we solved a regularization problem. Instead of considering Eq. (8), in [25] we introduced the following prior model on $\mathbf{X}$:

$$\mathrm{p}(\mathbf{X}|\mathbf{A},\nu) \propto \exp\left\{-\frac{\nu}{2}\|\mathbf{X}-\mathbf{AZ}\|_{\mathrm{F}}^2\right\}. \tag{10}$$

Notice that the constraint $\mathbf{X} = \mathbf{AZ}$ is only satisfied when $\nu \to \infty$. However, if ν is too high, the GP model parameters are not estimated well. To address this problem, we used in [25] the solution proposed in [33]. Unfortunately, the way ν should go to infinity depends on the problem and requires parameter tunning. Another solution, proposed in [32], is to fix ν; however for the pixel labeling problems it does not lead to good results. In this work we provide for the first time an estimation procedure for the precision parameter in Eq. (8).

We want to emphasize here the link between probabilistic modeling and ADMM optimization. We believe that this connection has not been sufficiently explored in the literature and could lead to the solution of very interesting and complex problems.

So far, we have used a GP classifier, but as Fig. 2 indicates we could utilize any other classifier instead, such as, Logistic Regression (LR). When LR is used we have a (simple) neural network with one convolutional layer [22]. Filter and LR coefficients can be estimated at the same time using backward propagation. Notice that we can also consider more elaborate neural networks. This will be explored in the experimental section and the methods will be named Logistic Regression Filtering (LRF) and Neural Networks Filtering (NNF).

We would like to mention that the filtering/classification problem has also been recently addressed in [24], that is, a

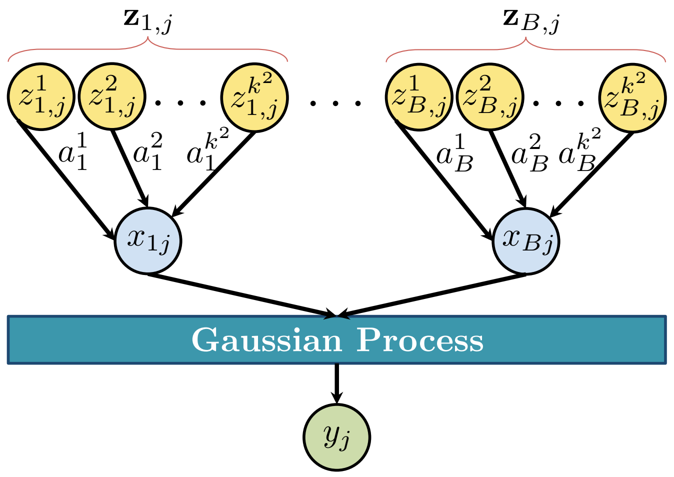


Fig. 2. Representation of filtering and GP classification system.

Support Vector Machine (SVM) is used instead of a GP in Fig. 2. The authors define the objective function of the SVM on $\mathbf{AZ}$ (instead of $\mathbf{Z}$), and add a regularization term on the filter coefficients $\mathbf{A}$. They develop an iterative algorithm based on gradient descent to optimize the objective function. This method is named Support Vector Machine Filtering (SVMF) in our experiments.

The use of GPs, and in general probabilistic classification models, has advantages over non-probabilistic ones. All its parameters can be automatically estimated from the data, while SVMs require manual parameter tuning. As we will see in the experimental section, this significantly increases the computational cost of the method proposed in [24]. Furthermore, while GP and SVM classifiers use kernel functions to solve non-linear classification problems, NN classifiers need to select the architecture which, together with the activation function, will provide a solution to the non-linear problems. This is usually a difficult task and must be performed by expert users. In summary, GP allows us to develop, based on a sound framework, an algorithm capable of addressing filtering and non-linear classification problems, whose parameters are automatically estimated from the samples.

IV. Variational Inference

In the Bayesian framework, if we knew the scaled Lagrange multipliers $\mathbf{U}$ for our filtering/labeling problem our goal would be to find

$$\mathrm{p}(\Theta|\mathbf{y},\mathbf{U}) = \frac{\mathrm{p}(\mathbf{y},\mathbf{U},\Theta)}{\mathrm{p}(\mathbf{y},\mathbf{U})}. \tag{11}$$

However, since $\mathrm{p}(\mathbf{y},\mathbf{U})$ can not be calculated, this posterior distribution must be approximated. In this work we use Variational inference (see [34] for details and application areas), and approximate the posterior by the distribution

$$\hat{\mathrm{q}}(\Theta) = \arg\min_{\mathrm{q}(\Theta)} \mathrm{KL}(\mathrm{q}(\Theta)\|\mathrm{p}(\Theta|\mathbf{y},\mathbf{U})), \tag{12}$$

where the Kullback-Leibler divergence is given by

$$\mathrm{KL}(\mathrm{q}(\Theta)\|\mathrm{p}(\Theta|\mathbf{y},\mathbf{U})) = \int \mathrm{q}(\Theta)\log\left(\frac{\mathrm{q}(\Theta)}{\mathrm{p}(\Theta|\mathbf{y},\mathbf{U})}\right)d\Theta = \int \mathrm{q}(\Theta)\log\left(\frac{\mathrm{q}(\Theta)}{\mathrm{p}(\mathbf{y},\mathbf{U},\Theta)}\right)d\Theta + \mathrm{const.} \tag{13}$$

This quantity is always non negative, and equal to zero, if and only if, $\mathrm{q}(\Theta)$ and $\mathrm{p}(\Theta|\mathbf{y},\mathbf{U})$ coincide.

Since $\mathrm{q}(\Theta)$ is too general, we need to include constraints on it. In this paper we use the mean field approximation [35] and constrain the posterior to be of the form

$$\mathrm{q}(\Theta) = \mathrm{q}(\mathbf{f})\mathrm{q}(\mathbf{X})\mathrm{q}(\boldsymbol{\alpha},\beta,\gamma)\prod_{i=1}^{B}\mathrm{q}(\mathbf{a}_i). \tag{14}$$

We additionally constrain $\mathrm{q}(\mathbf{X})$ and $\mathrm{q}(\boldsymbol{\alpha},\beta,\gamma)$ to be degenerate.

Due to the form of the joint likelihood defined in Eq. (4), the KL divergence cannot be evaluated. To solve this problem we bound the joint likelihood in Eq. (4), using the variational lower bound [31] [36]

$$\ln(1+e^{v}) \le \lambda(\xi)(v^2-\xi^2) + \frac{v-\xi}{2} + \ln(1+e^{\xi}), \tag{15}$$

where $\lambda(\xi) = \frac{1}{2\xi}\left(\frac{1}{1+e^{-\xi}} - \frac{1}{2}\right)$. Thus we have

$$\mathrm{p}(\mathbf{y}|\mathbf{f}) \ge \exp\left\{(\mathbf{y}-\frac{1}{2}\mathbf{1})^T\mathbf{f} - \mathbf{f}^T\Lambda\mathbf{f}\right\} \times \exp\left\{\boldsymbol{\xi}^T\Lambda\boldsymbol{\xi} + \frac{1}{2}\mathbf{1}^T\boldsymbol{\xi}\right\}\prod_{j=1}^{N}\boldsymbol{\sigma}(-\xi_j) = \mathbf{F}(\mathbf{y},\mathbf{f},\boldsymbol{\xi}), \tag{16}$$

where $\boldsymbol{\xi} = (\xi_1,\ldots,\xi_N)^T$, $\Lambda = \mathrm{Diag}(\lambda(\xi_1),\ldots,\lambda(\xi_N))$ and $\boldsymbol{\sigma}$ is the sigmoidal function, and $\mathbf{1}$ denotes the vector of all 1's. The inequality in Eq. (16) leads to the following lower bound for the joint probability distribution:

$$\mathrm{p}(\mathbf{y},\mathbf{U},\Theta) \ge \mathbf{M}(\mathbf{y},\mathbf{U},\Theta,\boldsymbol{\xi}) = \mathbf{F}(\mathbf{y},\mathbf{f},\boldsymbol{\xi})\mathrm{p}(\mathbf{U}|\mathbf{A},\mathbf{X},\beta)\mathrm{p}(\mathbf{f}|\mathbf{X},\gamma)\mathrm{p}(\mathbf{A}|\boldsymbol{\alpha}). \tag{17}$$

Finally, the KL divergence in Eq. (13) is majorized by

$$\mathrm{KL}(\mathrm{q}(\Theta)\|\mathrm{p}(\Theta|\mathbf{y},\mathbf{U})) \le \mathrm{KL}(\mathrm{q}(\Theta)\|\mathbf{M}(\mathbf{y},\mathbf{U},\Theta,\boldsymbol{\xi})) + \mathrm{const.} \tag{18}$$

Although a new set of unknowns $\boldsymbol{\xi}$ has been introduced, which must be estimated, now the KL divergence between $\mathrm{q}(\Theta)$ and $\mathbf{M}(\mathbf{y},\mathbf{U},\Theta,\boldsymbol{\xi})$ is mathematically tractable, and it can be used to approximate the posterior distribution by $\mathrm{q}(\Theta)$.

The solution to this optimization problem is given by [31]

$$\log\hat{\mathrm{q}}(\theta) = \mathbb{E}_{\mathrm{q}(\Theta_\theta)}\left[\log\mathbf{M}(\mathbf{y},\mathbf{U},\Theta,\boldsymbol{\xi})\right] \tag{19}$$

where $\theta\in\Theta$ and $\Theta_\theta = \Theta\setminus\theta$.

A. *Estimation of* $\mathrm{q}(\mathbf{f}),\mathrm{q}(\mathbf{A})$ *and* $\mathrm{q}(\mathbf{X})$

Since $\mathbb{E}_{\mathrm{q}(\Theta_\mathbf{f})}\left[\log\mathbf{M}(\mathbf{y},\mathbf{U},\Theta,\boldsymbol{\xi})\right]$ is a quadratic function on $\mathbf{f}$, its posterior distribution approximation is a Gaussian distribution with mean and covariance matrix given by respectively

$$<\!\mathbf{f}\!> = \Sigma_\mathbf{f}\left(\mathbf{y}-\frac{1}{2}\mathbf{1}\right), \quad \Sigma_\mathbf{f} = \left(\gamma\mathbf{K}^{-1}+2\Lambda\right)^{-1}. \tag{20}$$

Notice that, depending on $\mathbf{X}$, $\mathbf{K}$ can be singular. To overcome this problem we follow the approach proposed in [27]. Using the Sherman-Morrison-Woodbury identity, we have

$$\Sigma_\mathbf{f} = \gamma^{-1}\mathbf{K} - \gamma^{-2}\mathbf{KWBWK}, \tag{21}$$

where $\mathbf{W} = (2\Lambda)^{1/2}$ and $\mathbf{B} = (\mathbf{I}+\gamma^{-1}\mathbf{WKW})^{-1}$.

Since $\mathbb{E}_{\mathrm{q}(\Theta_{\mathbf{a}_i})}\left[\log\mathbf{M}(\mathbf{y},\mathbf{U},\Theta,\boldsymbol{\xi})\right]$, $i = 1,\ldots,B$, are quadratic functions on $\mathbf{a}_i$, the posterior distribution approximation for each filter is a Gaussian distribution with parameters

$$<\!\mathbf{a}_i\!> = \Sigma_i\beta\mathbf{Z}_i(\mathbf{U}_{i\cdot}+\mathbf{X}_{i\cdot})^T, \quad \Sigma_i = \left(\beta\mathbf{Z}_i\mathbf{Z}_i^T+\alpha_i\mathbf{I}\right)^{-1}, \tag{22}$$

where $\mathbf{X}_{i\cdot}$ and $\mathbf{U}_{i\cdot}$ denote the i-th rows of $\mathbf{X}$ and $\mathbf{U}$, respectively.

We derive the estimation of $\mathbf{X}$ for two widely used kernels, linear and Gaussian. Since we assume that $\mathrm{q}(\mathbf{X})$ is degenerate, in both cases we have to maximize $\mathbb{E}_{\mathrm{q}(\Theta_\mathbf{X})}\left[\log\mathbf{M}(\mathbf{y},\mathbf{U},\Theta,\boldsymbol{\xi})\right]$ with respect to $\mathbf{X}$. Taking derivatives with respect to $\mathbf{X}$, and equating them to 0 we obtain for the linear kernel

$$\mathbf{X}^T = \left(\gamma\mathbf{v}\mathbf{v}^T - \gamma^{-1}\mathbf{S} - \beta\mathbf{I}\right)^{-1}\beta(\mathbf{U}^T - \mathbf{Z}^T<\!\mathbf{A}\!>^T), \tag{23}$$

where $\mathbf{v} = \mathbf{K}^{-1}<\!\mathbf{f}\!>$ and $\mathbf{S} = \mathbf{WBW}$. For the Gaussian kernel we have

$$x_{ij} = \frac{\sum_{t\neq j}^{N} e_{tj}x_{it}\left(\frac{\gamma}{\sigma^2}v_jv_t - \frac{1}{\gamma\sigma^2}s_{tj}\right) + \beta(\mathbf{a}_i^T\mathbf{z}_{i,j} - u_{ij})}{\sum_{t\neq j}^{N} e_{tj}\left(\frac{\gamma}{\sigma^2}v_jv_t - \frac{1}{\gamma\sigma^2}s_{tj}\right) + \beta}, \tag{24}$$

where σ^2 is the Gaussian kernel parameter, v_j, s_{tj} and u_{ij} are entries of $\mathbf{v},\mathbf{S}$, and $\mathbf{U}$ respectively, and $e_{tj} = \exp\left\{-\frac{1}{2\sigma^2}\|\mathbf{x}_t - \mathbf{x}_j\|^2\right\}$.

Notice that to calculate $\mathbf{v}$, $\mathbf{K}^{-1}$ is again required. Using Eqs. (20), (21) we have

$$\mathbf{v} = \mathbf{K}^{-1}<\!\mathbf{f}\!> = \left(\gamma^{-1}\mathbf{I} - \gamma^{-2}\mathbf{SK}\right)\left(\mathbf{y}-\frac{1}{2}\mathbf{1}\right), \tag{25}$$

which overcomes the possible singularity of $\mathbf{K}$.

B. *Parameter estimation*

Since $\mathrm{q}(\boldsymbol{\alpha},\beta,\gamma)$ is a degenerate distribution, the estimation of $\boldsymbol{\alpha},\beta,\gamma$ and $\boldsymbol{\xi}$ are obtained by maximizing $\mathbb{E}_{\mathrm{q}(\mathbf{f})\mathrm{q}(\mathbf{A})\mathrm{q}(\mathbf{X})}\left[\log\mathbf{M}(\mathbf{y},\mathbf{U},\Theta,\boldsymbol{\xi})\right]$, which produces the following estimates

$$\alpha_i = \frac{K}{<\!\mathbf{a}_i\!>^T<\!\mathbf{a}_i\!> + \mathrm{Tr}\left[\Sigma_i\right]}, \tag{26}$$

$$\gamma = \frac{N}{<\!\mathbf{f}\!>^T\mathbf{v} + \mathrm{Tr}[\mathbf{K}^{-1}\Sigma_\mathbf{f}]}, \tag{27}$$

$$\xi_j = \sqrt{<\!f_j\!>^2 + \Sigma_\mathbf{f}(j,j)}. \tag{28}$$

C. Pseudo-observations and associated precision updates

Once all of the above distributions have been updated we need to recalculate the pseudo-observations, $\mathbf{U}$, in Eq. (8) and their associated precision parameter, β.

To provide insight on how these parameters are updated, let us consider the following optimization problem

$$\begin{array}{l}\min_{\mathbf{A},\mathbf{X}}\ \mathbf{M}_1(\mathbf{A})+\mathbf{M}_2(\mathbf{X})\\ \text{s.t.}\ \ \mathbf{X}=\mathbf{A}\mathbf{Z},\end{array} \tag{29}$$

where

$$\mathbf{M}_1(\mathbf{A})=\frac{1}{2}\sum_{i=1}^{B}\alpha_i\|\mathbf{a}_i\|_2^2, \tag{30}$$

$$\mathbf{M}_2(\mathbf{X})=\frac{1}{2}\log|\mathbf{K}|+\frac{\gamma}{2}{<}\mathbf{f}{>}^T\mathbf{K}^{-1}{<}\mathbf{f}{>}+\frac{\gamma}{2}\operatorname{Tr}[\mathbf{K}^{-1}\Sigma_{\mathbf{f}}], \tag{31}$$

and again the dependency of $\mathbf{K}$ on $\mathbf{X}$ has not been made explicit for ease of notation.

To solve the optimization problem in Eq. (29) using ADMM, we form the augmented Lagrangian

$$\begin{aligned}\mathcal{L}_\beta(\mathbf{A},\mathbf{X},\mathbf{L})=&\ \mathbf{M}_1(\mathbf{A})+\mathbf{M}_2(\mathbf{X})\\ &+\operatorname{Tr}[\mathbf{L}^T(\mathbf{X}-\mathbf{A}\mathbf{Z})]+\frac{\beta}{2}\|\mathbf{X}-\mathbf{A}\mathbf{Z}\|_{\mathrm{F}}^2\end{aligned} \tag{32}$$

and utilize the following iterative procedure

$$\begin{array}{ll}\mathbf{A}^{n+1} & :=\arg\min_{\mathbf{A}}\mathcal{L}_\beta(\mathbf{A},\mathbf{X}^n,\mathbf{L}^n)\\ \mathbf{X}^{n+1} & :=\arg\min_{\mathbf{X}}\mathcal{L}_\beta(\mathbf{A}^{n+1},\mathbf{X},\mathbf{L}^n)\\ \mathbf{L}^{n+1} & :=\mathbf{L}^n+\beta(\mathbf{X}^{n+1}-\mathbf{A}^{n+1}\mathbf{Z}).\end{array} \tag{33}$$

Using the scaled multipliers, $\mathbf{U}=\mathbf{L}/\beta$, we can write (see [26] chapter 3, for details)

$$\begin{aligned}\operatorname{Tr}[\mathbf{L}^T(\mathbf{X}-\mathbf{A}\mathbf{Z})]+&\frac{\beta}{2}\|\mathbf{X}-\mathbf{A}\mathbf{Z}\|_{\mathrm{F}}^2=\\ &\frac{\beta}{2}\|\mathbf{U}-(\mathbf{A}\mathbf{Z}-\mathbf{X})\|_{\mathrm{F}}^2-\frac{\beta}{2}\|\mathbf{U}\|_{\mathrm{F}}^2,\end{aligned} \tag{34}$$

and then the iterative procedure in Eq. (33) becomes

$$\begin{array}{ll}\mathbf{A}^{n+1} & :=\arg\min_{\mathbf{A}}\mathbf{M}_1(\mathbf{A})+\frac{\beta}{2}\|\mathbf{U}-(\mathbf{A}\mathbf{Z}-\mathbf{X})\|_{\mathrm{F}}^2\\ \mathbf{X}^{n+1} & :=\arg\min_{\mathbf{X}}\mathbf{M}_2(\mathbf{X})+\frac{\beta}{2}\|\mathbf{U}-(\mathbf{A}\mathbf{Z}-\mathbf{X})\|_{\mathrm{F}}^2\\ \mathbf{U}^{n+1} & :=\mathbf{U}^n-(\mathbf{A}^{n+1}\mathbf{Z}-\mathbf{X}^{n+1}).\end{array} \tag{35}$$

The scaled multipliers correspond to our pseudo observations in Eq. (8). Consequently, in the proposed estimation procedure all unknown posterior approximations are updated using these pseudo observations and then $\mathbf{U}$ is updated using the current filtered features and mean of the filter coefficients, that is,

$$\mathbf{U}^{new}=\mathbf{U}^{old}-({<}\mathbf{A}{>}^{new}\mathbf{Z}-{<}\mathbf{X}{>}^{new}). \tag{36}$$

Furthermore, the Bayesian framework allows us to estimate the associated precision parameter β without user tuning. Taking the derivative with respect to β of $\mathbb{E}_{\mathrm{q}(\mathbf{f})\mathrm{q}(\mathbf{A})\mathrm{q}(\mathbf{X})}\left[\mathbf{M}(\mathbf{y},\mathbf{U},\Theta,\boldsymbol{\xi})\right]$ and equating it to 0 we obtain

$$\beta^{new}=\frac{NB}{\|\mathbf{U}^{new}-({<}\mathbf{A}{>}^{new}\mathbf{Z}-\mathbf{X}^{new})\|_{\mathrm{F}}^2+T}, \tag{37}$$

where

$$T=\sum_{i=1}^{B}\operatorname{Tr}[\mathbf{Z}_i^T\Sigma_i\mathbf{Z}_i]. \tag{38}$$

We have experimentally observed (see Section VII), that the sequence of pseudo-observations in Eq. (36) converges to zero and that the β sequence in Eq. (37) increases to infinity. Both conditions guarantee that at convergence the estimated filter coefficients and the new obtained features satisfy

$${<}\mathbf{A}{>}\mathbf{Z}-{<}\mathbf{X}{>}=\mathbf{0}. \tag{39}$$

The whole estimation process is presented in Algorithm 1.

V. Algorithm Initialization

Since the KL divergence is a convex function of $\mathrm{q}(\theta)$ [37], for a given set of pseudo-observations $\mathbf{U}$ and precision parameter β, the convergence of Algorithm 1, without updating $\mathbf{U}$ and β, is guaranteed. However, $\mathbf{U}$ and β are updated at each iteration which implies that the objective function is not convex, and we cannot prove that Algorithm 1 always converges. For this reason, an important issue is the initialization of Algorithm 1, as different initializations can lead to different solutions. In this section we provide an initialization method which has led to satisfactory estimations of all the unknowns, and has not presented any convergence problems in the carried out experiments.

At convergence we would like for Algorithm 1 to provided filtered samples from different classes which are widely separated by the classifier. Observing that $\mathbf{X}$ is the link between the classifier and the filter estimation, we start with an $\mathbf{X}$ for which both classes are fully separated. Then the filter coefficients are estimated to transform $\mathbf{Z}$ into $\mathbf{X}$. In the experiments we have initialized each column of $\mathbf{X}^0$ to $\mathbf{1}\in\mathbb{R}^B$ for the samples in Class 1, and $-\mathbf{1}$ for the samples in Class 0.

For the classifier we start with $\mathrm{q}^0(\mathbf{f})\sim\mathcal{N}(\mathbf{0},\mathbf{I})$. Then we could use Eq. (27) to initialize γ^0; however, since $\mathbf{K}$ may be singular we use the following initialization $\gamma^0=N/(\sum_i d_i^{-1})$, where d_i are the non-zero eigenvalues of $\mathbf{K}$. We also use $\boldsymbol{\xi}^0=\mathbf{1}$. The filter coefficient distributions are initialized using $\mathrm{q}^0(\mathbf{a}_i)=\mathcal{N}(\mathbf{a}_i|\mathbf{0},\mathbf{I})$, and $\boldsymbol{\alpha}^0=\mathbf{1}$. We initialize the pseudo-observations using $\mathbf{U}^0=\mathbf{0}$ and finally utilize Eq. (37), with $\mathbf{X}^0,\mathrm{q}^0(\mathbf{a}_i)$ and $\mathbf{U}^0$ for the initialization of $\beta^0=NB/(\|\mathbf{X}^0\|_{\mathrm{F}}^2+\operatorname{Tr}[\mathbf{Z}^T\mathbf{Z}])$.

VI. Classification Rule

Given a new sample and its corresponding neighborhood $\mathbf{Z}^*$, we use the estimated filter to obtain its filtered version $\mathbf{x}^*={<}\mathbf{A}{>}\mathbf{Z}^*$. The estimated GP classifier allows us to calculate the probability of belonging to Class 1 using

$$\mathrm{p}(y^*=1|\mathbf{y})=\int\mathrm{p}(y^*=1|f^*)\mathrm{p}(f^*|\mathbf{y})\mathrm{d}f^*, \tag{40}$$

where $\mathrm{p}(y^*=1|f^*)=\boldsymbol{\sigma}(f^*)$ and

$$\mathrm{p}(f^*|\mathbf{y})=\int\mathrm{p}(f^*|\mathbf{f})\mathrm{p}(\mathbf{f}|\mathbf{y})\mathrm{d}\mathbf{f}\approx\int\mathrm{p}(f^*|\mathbf{f})\mathrm{q}(\mathbf{f})\mathrm{d}\mathbf{f}. \tag{41}$$

Algorithm 1 Gaussian Processes Filtering (GPF)

```
Require: $\mathbf{X}^0, \gamma^0, \boldsymbol{\xi}^0, \mathbf{A}^0, \boldsymbol{\alpha}^0, \mathbf{U}^0$ and $\beta^0$.
1: repeat
2:     Calculate $\mathrm{q}^{n+1}(\mathbf{f})$ using Eq. (20).
3:     Calculate classifier parameters $\gamma^{n+1}$ and $\boldsymbol{\xi}^{n+1}$ using Eq. (27) and (28) respectively.
4:     Calculate the filterbank posterior $\mathrm{q}^{n+1}(\mathbf{a}_i), i = 1, \dots, B$ using Eq. (22).
5:     Calculate the filterbank parameters $\boldsymbol{\alpha}^{n+1}$ using Eq. (26).
6:     Calculate $\mathbf{X}^{n+1}$ using Eq. (23) (Linear kernel) or Eq. (24) (Gaussian kernel).
7:     Calculate $\mathbf{U}^{n+1}$ using Eq. (36).
8:     Calculate $\beta^{n+1}$ using Eq. (37).
9: until Convergence
```

To calculate $\mathrm{p}(f^*|\mathbf{f})$ we notice that

$$\mathrm{p}(\mathbf{f}, f^*) \sim \mathcal{N}\left(\mathbf{0}, \left[\begin{array}{cc} \gamma^{-1}\mathbf{K} & \gamma^{-1}\mathbf{h} \\ \gamma^{-1}\mathbf{h}^T & \gamma^{-1}\mathbf{c} \end{array}\right]\right), \quad (42)$$

where $\mathbf{h}^T = (\mathbf{k}(\mathbf{x}^*, \mathbf{x}_1), \dots, \mathbf{k}(\mathbf{x}^*, \mathbf{x}_N))$ and $c = \mathbf{k}(\mathbf{x}^*, \mathbf{x}^*)$; from this distribution we obtain the following conditional distribution

$$\mathrm{p}(f^*|\mathbf{f}) = \mathcal{N}\left(f^*|\mathbf{h}^T\mathbf{K}^{-1}\mathbf{f}, \gamma^{-1}(c - \mathbf{h}^T\mathbf{K}^{-1}\mathbf{h})\right). \quad (43)$$

Using the above equation in Eq. (41) we obtain

$$\mathrm{p}(f^*|\mathbf{y}) = \mathcal{N}\left(f^*|\mathbf{h}^T\mathbf{v}, \gamma^{-1}c - \gamma^{-2}\mathbf{h}^T\mathbf{S}\mathbf{h}\right). \quad (44)$$

Finally using the approximation in [31], (chapter 4) we have

$$\mathrm{p}(y^* = 1|\mathbf{y}) \approx \boldsymbol{\sigma}\left(\kappa(\gamma^{-1}c - \gamma^{-2}\mathbf{h}^T\mathbf{S}\mathbf{h})\mathbf{h}^T\mathbf{v}\right), \quad (45)$$

where $\kappa(s) = (1 + \pi s/8)^{-1/2}$. Thus the following classification rule is obtained

$$y^* = \left\{\begin{array}{lll} 1 & \text{if} & \mathrm{p}(y^* = 1|\mathbf{y}) \geq 0.5 \\ 0 & \text{if} & \mathrm{p}(y^* = 1|\mathbf{y}) < 0.5 \end{array}\right. . \quad (46)$$

Notice that since we can calculate the probability of belonging to a given class, it is easy to extend the proposed method to multiclass problems by using an *one-vs-all* approximation. For each new sample, its probability of belonging to each class is calculated, and then, the sample is assigned to the class with maximum probability.

VII. Experimental Results

In this section we show how the proposed method, henceforth denoted by GPF, works on synthetic and real data sets. GPF was implemented in MATLAB© and it is available at http://decsai.ugr.es/vip/resources/GPF.html. We compare it with the methods described in Section III: GPC [27], LRF [22], SVMF [24] and NNF [22]. In the experiments we have used for SVMF its MATLAB© implementation available in the website of the authors. For GPC we use the classification part of the method developed here, and for LRF and NNF we use the Neural Networks Toolbox of MATLAB©.

A main drawback of SVMF, is that its parameters must be estimated ahead of time. In order to perform a fair comparison, the SVMF parameters are set using 5-fold cross-validation. For NNF, in addition to the first convolutional layer, we consider four fully connected layers with $B+4$ nodes in each layer. The neural networks for LRF and NNF were initialized using the Nguyen-Widrow method, trained using Levenberg-Marquadt algorithm, and sigmoidal functions were used as activation functions. Notice that with GPF we estimate all the model parameters as well as the latent variables. All experiments were run on an Intel i7@2.80 GHz processor.

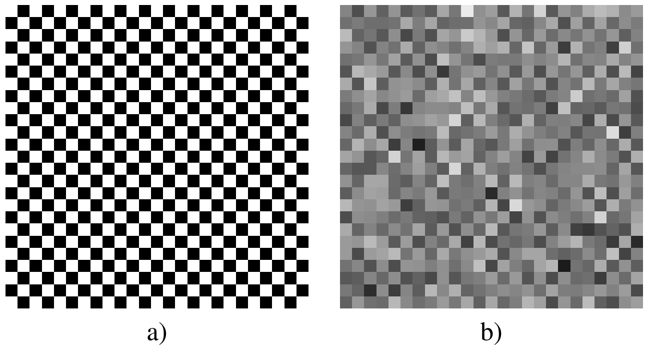


Fig. 3. a) 25×25 zoomed groundtruth in synthetic experiment, b) 25×25 zoomed observation in synthetic experiment.

A. Synthetic Experiment

For the first experiment we use the following synthetic dataset: the groundtruth is an image of size 100×100 pixels, where black and white pixels alternate in a chess table pattern. A zoomed version of the groundtruth is shown in Fig. 3a. Classes $\mathcal{C}_0$ and $\mathcal{C}_1$ denote black and white pixels, respectively.

Given the groundtruth, features are generated as independent realizations from a $\mathcal{N}(0.25, \sigma^2)$ and $\mathcal{N}(0.75, \sigma^2)$, for classes $\mathcal{C}_0$ and $\mathcal{C}_1$, respectively. In Fig. 3b we see the zoomed observed features for $\sigma = 0.4$. Notice that there are pixels which cannot be easily classified; we therefore need neighboring information to perform a successful classification.

We start by studying the influence of the training set size on the final performance of our algorithm. We set $\sigma = 0.4$ and consider training set sizes $N = 8, 16, 24, 32, 40$ with the same number of samples for each class. For the test set we randomly selected 1000 samples for each class. The experiments were repeated 10 times. In Fig. 4 we plot the obtained mean Overall Accuracies (OA) for varying filter and training set sizes. Notice that for 7×7 and 9×9 sizes, the mean OA is always 100%. For 3×3 and 5×5 sizes the curves increase until they reach the maximum at $N = 32$, and then they decrease slightly for $N = 40$.

Next, we study the effect of noise on the performance of the algorithm. We generated datasets for $\sigma = \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$ and performed training and testing randomly selecting 20 and 1000 pixels from each class, respectively. The experiments were repeated 10 times. In Fig. 5 we observe that for $\sigma = 0.2$ GPF obtains 100% mean OA for all filter sizes. However, all curves decrease with higher noise. Notice that the values of OA decrease faster for smaller filter sizes.

TABLE I
OVERALL ACCURACIES (%) AND MEAN TIMES (SECONDS) FOR SYNTHETIC EXPERIMENT

		3×3					5×5					7×7					9×9				
Realizations	**GPC**	**NLM**	**LRF**	**NNF**	**SVMF**	**GPF**	**NLM**	**LRF**	**NNF**	**SVMF**	**GPF**	**NLM**	**LRF**	**NNF**	**SVMF**	**GPF**	**NLM**	**LRF**	**NNF**	**SVMF**	**GPF**
1	74.20	92.60	87.35	95.20	95.60	**96.90**	97.40	98.80	**99.95**	99.15	**99.95**	99.45	99.60	**100**	99.15	**100**	99.85	**100**	**100**	99.25	**100**
2	73.45	93.10	90.85	90.10	95.00	**95.60**	96.70	95.90	99.20	98.80	**99.85**	98.70	99.00	99.95	99.00	**100**	99.40	99.95	**100**	99.20	**100**
3	72.05	90.55	90.35	**96.10**	95.35	96.05	96.25	97.20	99.40	98.80	**99.95**	98.20	99.60	**100**	99.05	**100**	99.15	**100**	99.95	99.20	**100**
4	72.15	90.95	91.80	95.45	95.05	**95.80**	96.45	97.30	99.25	99.35	**99.85**	98.85	95.70	**100**	99.40	**100**	99.50	**100**	99.95	99.35	**100**
5	73.80	92.95	85.50	92.45	95.50	**96.15**	97.30	97.60	**99.70**	98.70	**99.70**	98.75	97.55	**100**	99.20	**100**	99.55	**100**	99.95	99.25	**100**
6	73.35	91.45	88.80	93.55	93.80	**95.55**	96.80	93.75	99.30	98.85	**99.70**	98.50	99.00	**100**	99.35	**100**	99.35	**100**	**100**	99.25	**100**
7	74.20	93.35	92.40	91.95	95.65	**96.25**	98.15	99.15	99.90	98.50	**99.95**	99.40	99.95	**100**	97.90	**100**	99.75	**100**	**100**	98.90	**100**
8	74.25	93.60	94.60	94.30	96.20	**97.00**	97.40	98.70	99.90	99.15	**99.95**	98.95	99.15	**100**	99.05	**100**	99.45	**100**	**100**	98.60	**100**
9	72.85	91.40	92.35	95.30	95.65	**96.15**	96.35	95.30	99.85	99.65	**99.90**	98.50	97.40	98.40	97.80	**100**	99.15	99.95	**100**	99.30	**100**
10	73.10	92.15	93.85	94.95	94.80	**96.15**	97.30	97.20	**99.85**	99.30	**99.85**	99.15	97.15	**100**	99.60	**100**	99.65	99.90	**100**	99.35	**100**
Mean	73.34	92.21	90.78	93.93	95.26	**96.16**	97.01	97.09	99.63	99.025	**99.865**	98.84	98.41	99.83	98.95	**100**	99.48	99.98	99.98	99.165	**100**
Mean Times	0.178	101	0.26	0.29	13.132	0.02	106	0.21	0.28	15.725	0.041	106	0.22	0.29	18.857	0.105	106	0.21	0.27	22.733	0.199

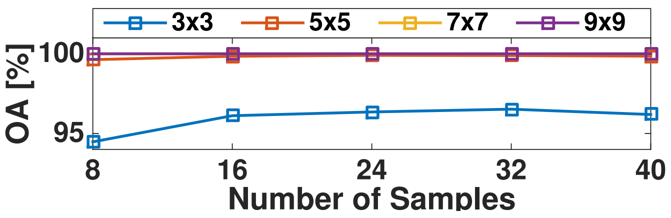


Fig. 4. Mean OAs for GPF varying training set and neighborhood sizes in the synthetic experiment.

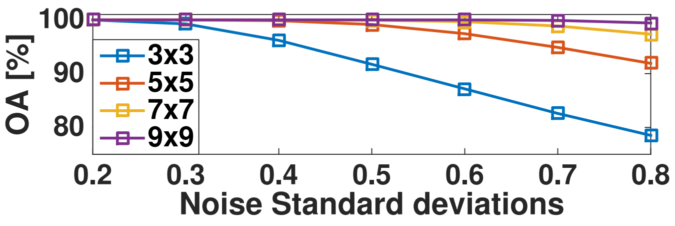


Fig. 5. Mean OAs for GPF with different noise deviations and filter sizes in the synthetic experiment.

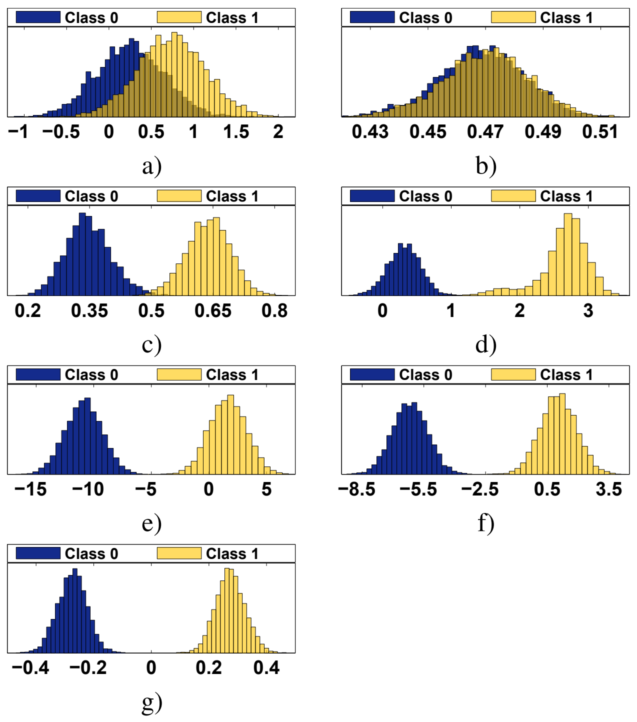


Fig. 6. Synthetic experiment: a) Histogram of the original test samples. Histogram of the filtered test samples with: b) a mean filter, c) the NLM filter, d) the SVMF filter, e) the LRF filter, f) the NNF filter, g) the GPF filter.

To compare GPF to other methods, σ is set to 0.4. Training and testing were performed using 20 and 1000 randomly selected pixels from each class, respectively, and the experiments were repeated 10 times.

In addition to GPC, LRF, NNF, and SVMF, we also preprocessed the image using a Non-Local Mean (NLM) filter for denoising, and applied GPC to the filtered image. The real value of σ was provided to NLM as input parameter. However, notice that normally σ will be unknown.

Table I shows the performance of GPF, GPC, NLM, LRF, NNF, and SVMF on the 10 realizations. Filters of 4 different sizes: 3×3, 5×5, 7×7 and 9×9 were considered. GPF obtained a mean OA of 96.16%, 99.87%, 100%, 100% for 3×3, 5×5, 7×7, and 9×9 filter sizes, respectively. In all the realizations the OA was greater than 95% and for large enough filters (7×7 and 9×9), no errors were made in the test phase. The GPC mean OA was 73.3%, that is approximately 22% less than the worst GPF OA value. This result demonstrates that using only the intensity value of a given pixel, it is not sufficient to correctly classify samples in our synthetic example; the classifier must make use of the neighboring information to perform well. The remaining methods use this information. For 3×3, 5×5, 7×7 and 9×9 filter sizes, the obtained mean OAs for NLM were 92.21%, 97.01%, 98.84%, 99.48%, respectively, for LRF they were 90.78%, 97.09%, 98.41%, 99.98%, respectively, for NNF the mean OAs were 93.93%, 99.63%, 99.83%, 99.98%, respectively. Finally for SVMF the obtained mean OAS were 95.26%, 99.025%, 98.95%, 99.165% for the mentioned above filter sizes respectively. In all cases, GPF obtained the best mean OA, followed be NNF and SVMF, while NLM and LRF close the ranking. It is interesting to note that although NLM does not calculate an optimal filter for classification, its performance is close to LRF (even outperforms it for the 3×3 case).

Let us graphically analyze the obtained results. In Fig. 6a we plot the feature histograms of the test samples for both classes.

Since both histograms overlap substantially, this explains the low GPC performance. A classic pre-processing step in an image classification problem such as the one considered here, is to filter the original image with a mean filter. In Fig. 6b we plot the histograms of the mean filtered test samples. As the two histograms are almost fully overlapped, this will render the classification task impossible for any classifier. It is an example where using a non-optimal filter leads to a harder classification problem. Figs. 6c and 6d show the histograms of the filtered test samples using the 9×9 filters provided by NLM and SVMF, respectively, for the 6th realization. The NLM histograms are almost separated, however there are a small set of samples around 0.5 that NLM filter could not separate. The same applies to SVMF, both histograms are almost separated; however samples in class $\mathcal{C}_1$ spread more than samples in class $\mathcal{C}_0$, and some bins in the middle overlap. Finally, Figs. 6e-g show the histograms when the test samples were filtered using the 9×9 filters obtained by LRF, NNF and GPF for realization 6. As it can be observed the histograms are completely separated.

Let us now examine the training time for each method. We observe in Table I that the GPF mean times for the 10 realizations were 0.02, 0.04, 0.11, and 0.2 secs. for 3×3, 5×5, 7×7, and 9×9 filter sizes, respectively. GPC needed on the average 0.17 seconds; this time is comparable to the GPF 9×9 case. Notice that although GPF has more unknowns to estimate, it was even faster than GPC for small filters. It is due to the fact that at each iteration, the classes become more separated, and GPF needed fewer iterations to converge than GPC. NLM required around 100 secs. for training, because it previously has to filter the image in a highly time consuming way. The computation time for LRF and NNF is around 0.2 secs. for all the filter sizes, which is comparable to the worst GPF time. For SVMF, the computation times were between 13.13 seconds for filters of size 3×3 and 22.73 for filters of size 9×9. SVMF required considerable time to obtain the values of the parameters by cross-validation.

At the top row of Fig. 7 we plot the 3×3, 5×5, 7×7 and 9×9 filters estimated by GPF at realization 6. We observe that the coefficients were larger for pixels belonging to the same class as the central pixel, while they were smaller for pixels belonging to the other class. When the observations are filtered using this filter we obtain the images in the middle row of Fig. 7. Pixels in class $\mathcal{C}_1$ output high values while pixels in class $\mathcal{C}_0$ smaller ones. Finally at the bottom of Fig. 7 we plot the classification maps for each classifier. Notice that there are some mistakes for 3×3 and 5×5 filter sizes.

In Fig. 8a we plot the Frobenius norm of the scaled multipliers $\mathbf{U}$ for each iteration of Algorithm 1, for filter size 3×3. It reaches the maximum at first iteration, and decreases until zero after 8 iterations. In Fig. 8b we plot the values of parameter β. It starts with a small value, which is increased at each iteration of algorithm 1. At convergence, the Frobenius norm of $\mathbf{U}$ is zero, and β reaches high values, which implies that the constraint $\mathbf{X} = \mathbf{AZ}$ is fulfilled.

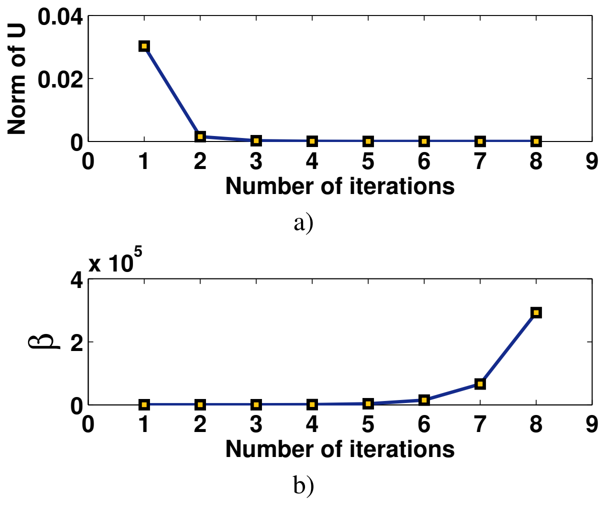


Fig. 8. Synthetic experiment: a) $||\mathbf{U}||_\mathrm{F}$ at each iteration of Algorithm 1, b) values of β at each iteration of Algorithm 1.

B. Multispectral Two-Class Image Classification

For the second experiment we used a multispectral image of the city of Rome (Italy). This image was captured by ERS2 SAR and Landsat TM sensors in 1995. The goal in this classification problem is to distinguish between Urban (Class $\mathcal{C}_1$) and Non-Urban (Class $\mathcal{C}_0$) pixels.

The image was acquired in the context of the Urban Expansion Monitoring project [30]. It is of size 1440×930 pixels and has 10 bands: 2 SAR backscattering intensities captured with 35 days difference, the correlation between them which is called the interferometric coherence, and the 7 multispectral bands of Landsat TM sensor. In Fig. 10a we show a small part (400×200 pixels) of the RGB bands of the image. A reference land cover map provided by the Italian Institute of Statistics (ISTAT) is also available, which was used for training and also to measure the classifier performance. The reference map is shown in Fig. 10b where yellow pixels belong to class urban, blue pixels to class no-urban and red pixels have unknown class.

For training, 100 pixels were randomly selected (50 samples from each class) and 2000 pixels were used as the test set (1000 for each class). The experiment was repeated 10 times with different training sets.

Table II shows the OA obtained in all realizations and the mean OA for the compared methods. In all cases, GPF obtained OA higher than 93%, and mean OA of 94.25%, 94.77%, 94.70%, and 94.63% for 3×3, 5×5, 7×7, and 9×9 filter sizes, respectively. For GPC, the mean OA was 90.71%, which implies approximately 4% worse performance than GPF. The mean OA for LRF was 81.06%, 81.59%, 82.25%, and 82.09% for 3×3, 5×5, 7×7, and 9×9 filter sizes, respectively. The mean OA for NNF was 90.47%, 91.72%, 90.02%, and 90.87% for 3×3, 5×5, 7×7, and 9×9 filter sizes, respectively. The mean OA for SVMF was 92.96%, 93.25%, 94.31%, and 94.26% for 3×3, 5×5, 7×7, and 9×9 filter sizes, respectively. Although SVMF obtained

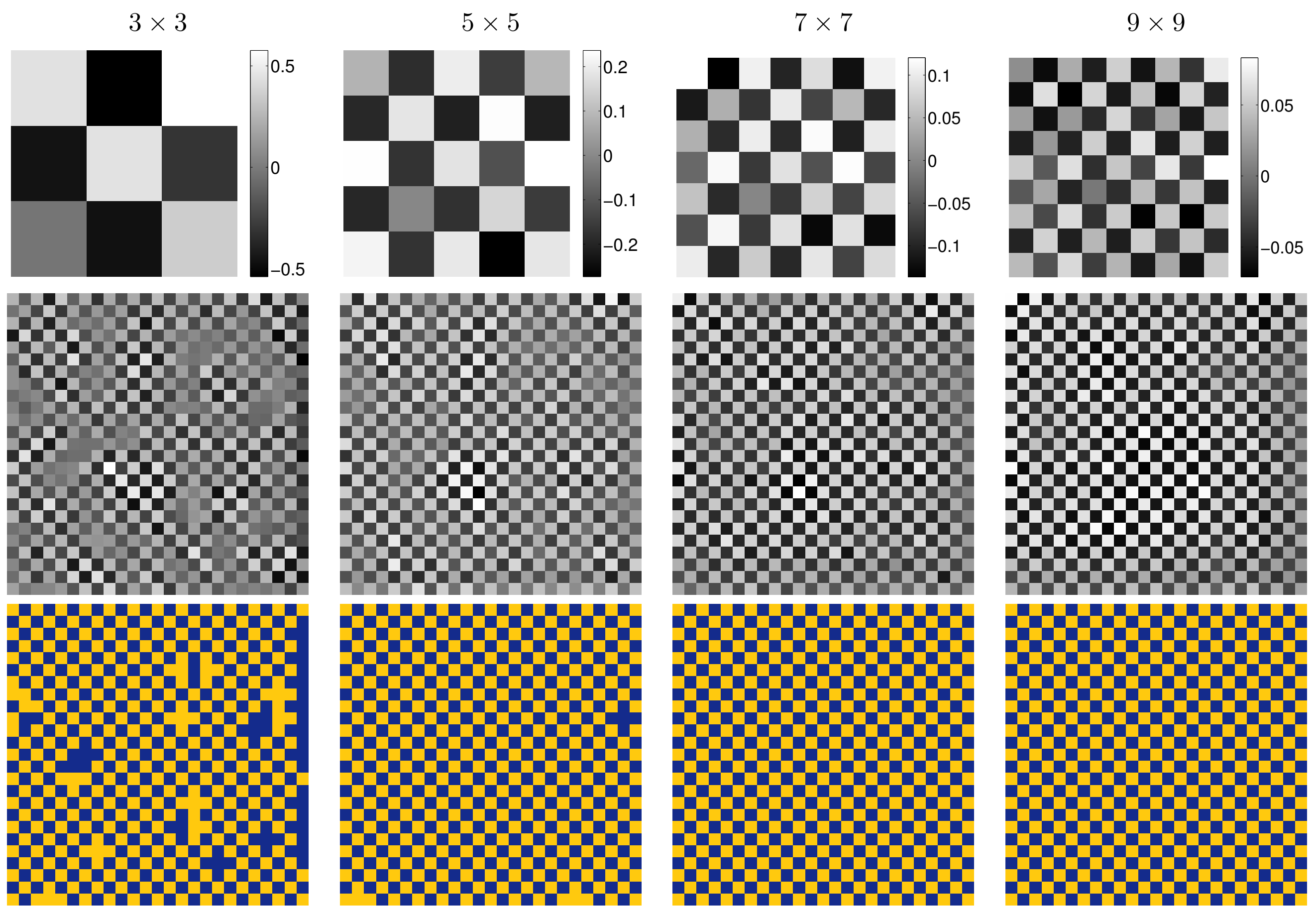


Fig. 7. Synthetic experiment: Top row shows the obtained filters for the different filter sizes. Middle row shows the filtered images obtained by filtering the observation with filters at the top row. Bottom row shows the classification maps in each case where yellow represents Class 1 and blue Class 0.

better results in some cases, GPF obtained the best mean OA in all cases.

In Table II we also show for each method the mean times of the training phase. The fastest method was GPC with 0.41 seconds. GPF needed 5.80, 6.66, 10.67, and 20.96 secs., for 3×3, 5×5, 7×7, and 9×9 filter sizes, respectively. LRF needed, for training, smaller times than the other methods: 2.40, 2.41, 2.52 and 2.76 secs. for 3×3, 5×5, 7×7, and 9×9 filter sizes, respectively, while NNF needed 3.70, 3.56, 4.04 and 4.77 secs. for 3×3, 5×5, 7×7, and 9×9 filter sizes, respectively. Again, due to the use of cross-validation for parameter selection, SVMF needed much more time for training: 185.95, 490.58, 948.72, and 1569.94 secs., for 3×3, 5×5, 7×7, and 9×9 filter sizes, respectively.

We conclude that the proposed method is capable for obtaining better results than LRF, NNF and SMVF. Moreover although LRF and NNF were faster, GPF obtained much better accuracies on a reasonable amount of time.

We believe that our method outperforms the others because it takes into account the uncertainty of the estimates which can be very relevant when few samples are available. Notice, however, that when the training set is large, SVMF sometimes outperforms our method, see Table II.

In Fig. 9 we plot the GPF mean OA for each filter size. Unlike the synthetic experiment, the maximum OA is obtained for a 5×5 filter. In this case, when a pixel is far from the central pixel it does not provide useful information towards its classification, and therefore the OA decreases when the filter becomes larger.

Fig. 10c shows the original coherence band. Pixels corresponding to the city (urban pixels) have larger values than pixels belonging to class non-urban. That is, coherence is a very useful feature to distinguish between both classes. The coherence image is noisy, because ERS2 SAR images are corrupted by multiplicative noise [30], which is transferred to the coherence band when correlating both SAR bands. In [30] the authors utilized a 9×9 filter to reduce the noise in the coherence band. The obtained filtered coherence band image is shown in Fig. 10d. This image is smoother than the original one and the noise has been reduced.

To evaluate the classification methods, we randomly selected 100 pixels for training (50 from each class) and 2000 pixels for testing (1000 from each class) from the original and filtered coherence bands. These experiments were repeated 10 times with different training sets, and the obtained results were the following: GPC obtained a mean OA of 80.38% for the original coherence band; for LRF and NNF the mean OA values were 79.21% and 84.23%, respectively; this OA value reached 91.24% when using the filtered coherence band. The filters estimated by SMVF and GPF on the original

TABLE II
OVERALL ACCURACIES (%) AND MEAN TIMES (SECONDS) FOR MULTISPECTRAL TWO-CLASS IMAGE CLASSIFICATION PROBLEM.

		3×3				5×5				7×7				9×9			
Realizations	**GPC**	**LRF**	**NNF**	**SVMF**	**GPF**	**LRF**	**NNF**	**SVMF**	**GPF**	**LRF**	**NNF**	**SVMF**	**GPF**	**LRFF**	**NNF**	**SVMF**	**GPF**
1	91.05	86.65	90.60	93.85	**94.00**	79.15	93.75	92.70	**94.70**	88.95	89.75	93.55	**94.40**	81.45	91.60	**94.45**	94.15
2	88.50	80.95	90.05	93.75	**93.90**	77.85	89.85	93.20	**94.65**	82.70	86.05	**94.45**	93.80	80.10	92.60	93.45	**93.55**
3	92.45	78.35	88.95	93.80	**94.85**	82.20	92.45	93.25	**94.95**	77.50	90.25	93.50	**94.35**	81.55	89.90	93.55	**94.20**
4	90.35	76.85	90.45	94.35	**94.55**	82.40	90.85	94.95	**95.30**	78.65	89.20	94.80	**95.80**	77.25	89.40	94.65	**95.45**
5	90.60	82.95	90.30	89.75	**95.20**	80.30	93.25	89.75	**95.90**	80.00	93.35	94.20	**95.00**	80.05	94.25	94.25	**94.90**
6	90.00	87.05	90.80	**94.05**	93.20	84.75	89.95	93.95	**94.50**	85.65	86.75	94.40	**94.60**	84.45	85.05	**95.20**	94.85
7	90.90	76.90	91.30	91.25	**94.25**	82.60	90.65	93.60	**94.65**	87.65	89.65	94.05	**94.95**	88.50	90.70	93.95	**94.95**
8	91.45	79.05	91.55	92.15	**93.75**	77.90	90.85	**94.00**	93.85	86.55	93.20	**96.00**	94.20	81.10	92.15	**95.20**	94.55
9	89.90	86.85	91.20	92.65	**94.45**	84.90	93.00	93.15	**94.20**	79.90	91.20	94.15	**94.35**	83.30	88.65	94.60	**94.55**
10	91.85	75.00	89.50	94.00	**94.30**	83.90	92.65	93.90	**95.00**	75.00	90.85	93.95	**95.55**	83.15	94.40	93.25	**95.63**
Mean	90.71	81.06	90.47	92.96	**94.25**	81.59	91.72	93.25	**94.77**	82.25	90.02	94.31	**94.70**	82.09	90.87	94.26	**94.63**
Mean Times	0.41	2.40	3.70	185.95	5.80	2.41	3.56	490.58	6.66	2.52	4.04	948.72	10.67	2.76	4.77	1569.94	20.96

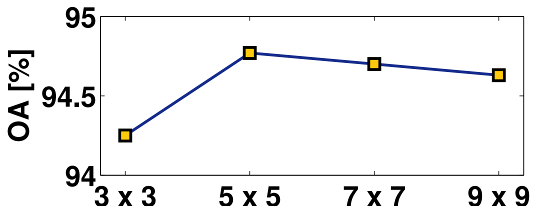


Fig. 9. Mean Overall Accuracies for different filter sizes, in the multispectral image classification problem.

coherence band performed better with 93.5% and 93.64% mean OA, respectively. In Fig. 10e, 10f, 10g, and 10h we show the filtered coherence band using the filters provided by LRF, NNF, SMVF and GPF for realization 9, respectively. Notice that LRF obtained an almost unrecognizable image, while NNF, SMVF and GPF better preserve some image details; notice, for instance, that the Tiber river was almost unnoticeable in Fig.10d, but it can be observed in 10f, 10g, and 10h.

For each method, and realization 9, in Fig. 11 we plot the obtained filters in both the spatial and Fourier domains (top and bottom rows, respectively). In the Fourier domain we observe that the resulting filter for LRF in Fig. 11a, concentrates most of its energy in high frequencies, while for NNF, SVMF and GPF in Figs. 11b, 11c and 11d, respectively, are low-pass filters. Notice that SVMF and GPF filters are very similar, and they also result in similar OAs.

C. Hyperspectral Multiclass Image Classification

In this experiment we evaluate the proposed method on a multiclass classification problem. The dataset is an hyperspectral image from the city of Pavia (Italy). This image was acquired by sensor DAIS9715, and has 40 bands in the spectral range $[0.5, 1.76]\mu m$, with high spatial resolution (5m pixel). In Fig. 12a we show bands $[8, 3, 1]$ in false RGB. In this case the goal is to solve a classification problem with 9 classes: WATER, MEADOWS, PARKING LOT, BITUMEN, ASPHALT, BARE SOIL, BRICK ROOFS, TREES, and SHADOWS. Fig. 12b shows the available groundtruth, which was used to select the training set and measure the classification performance.

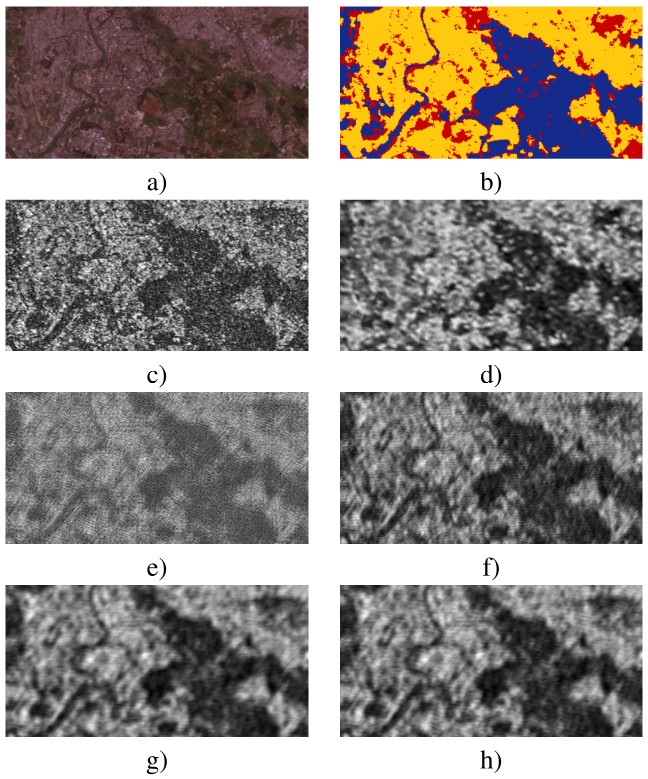


Fig. 10. Multispectral image classification experiment: a) Original RGB image, b) Available groundtruth with Urban (YELLOW), No-urban (BLUE) and unknown pixels (RED), c) Original coherence band, d) Filtered coherence band obtained in [30], e) Filtered coherence band obtained by LRF, f) Filtered coherence band obtained by NNF, g) Filtered coherence band obtained by SVMF, h) Filtered coherence band obtained by GPF.

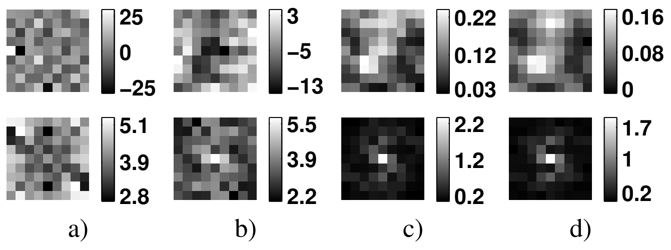


Fig. 11. Obtained filters for coherence band in multispectral image classification experiment. Top row corresponds to spatial domain and bottom row to Fourier domain. Columns correspond to the different methods: a) LRF, b) NNF, c) SVMF, and d) GPF.

To implement the multiclass GPF, we use the *one-vs-all* approximation explained in Section VI. The same approximation was also used for the multiclass GPC. For LRF and NNF we use a output layer of size 9 (number of classes), and a softmax activation function. For SVMF we use the multiclass *one-vs-all* approximation in [24].

For the training set, we randomly selected 270 samples (30 for each class) and 1800 samples for the test set (200 for each class), and the filter size was set to 3×3. The experiment was repeated 10 times to obtain unbiased results.

Table III shows the obtained OA for each method with 10 realizations. GPF achieved a mean OA of 96.94%; meanwhile GPC obtained 95.75% and LRF obtained 80.88%. Notice that LRF sometimes performs well, even it is the best method for realization 4. However, in some cases its OA is around 65%. We can conclude that LRF is very sensitive to the training set, and if it is not trained with a good set of samples its performance deteriorates. On the other hand NNF obtained a mean OA of 96.65%, a result very close to GPF performance. Surprisingly SVMF obtained 92.72%, approximately 3% less than GPC. This is due to the fact that in an *one-vs-all* approximation, SVMF calculates the distance of each sample to the decision boundary and the sample is assigned to the class with maximum distance. That is, unlike GPC, LRF, NNF and GPF, SVMF does not take into account the uncertainty during classification, which leads to a higher error rate.

The Confusion Matrix is a tool to demonstrate which classes are more difficult to distinguish for a classifier. Its columns represent the true classes while its rows the predictions, thus the (i, j)-th entry is the number of samples belonging to the j-th class, which the classifier assigns to i-th class. Fig. 13a shows the mean Confusion Matrix when GPC is applied. This matrix has 7 non-zero values outside the main diagonal. For GPF the mean Confusion matrix is shown in Fig. 13b, which only has 5 non-zero values outside the main diagonal. Notice that in comparison with the GPC Confusion Matrix, GPF does not make any classification errors for classes 3, 6, and 9, but it makes some mistakes for classes 2, 4 and 5. Notice the difference between the two methods for class 3 (BRICK ROOFS) shown in Fig. 14. We also show the Confusion Matrices obtained by the remaining methods in Figs. 13c, (SVMF) 13d (LRF) and 13e (NNF). In all these cases are more than 5 non-zero values outside the main diagonal.

Finally, for the brick roofs versus the rest, we have plotted the histograms for the obtained posterior probabilities of belonging to class brick roofs for the test samples. Bins close to 0 and 1 have many more elements (over 200), but they have been cut to better display the interesting part. In Fig. 14a, some elements not belonging to class 3 (brick roofs) have posterior probabilities of belonging to class brick roofs between 0.6 and 0.8. However when our GPF is used, all these samples have probabilities less than 0.5, see Fig. 14b.

VIII. Conclusion

In many image classification tasks, the image is first filtered to improve the separation between classes. However, a wrong filter selection can produce the opposite effect, i.e., make the classification harder. To solve this problem, in this work we have proposed a Bayesian model which combines the training of a Gaussian process classifier with an optimal filter bank estimation. To link both procedures, we have used the Alternating Direction Method of Multipliers by including a pseudo-observation model in our Bayesian framework. ADMM has allowed us to alternate between the estimation of classifier parameters and filter coefficients. Its convergence has been empirically established.

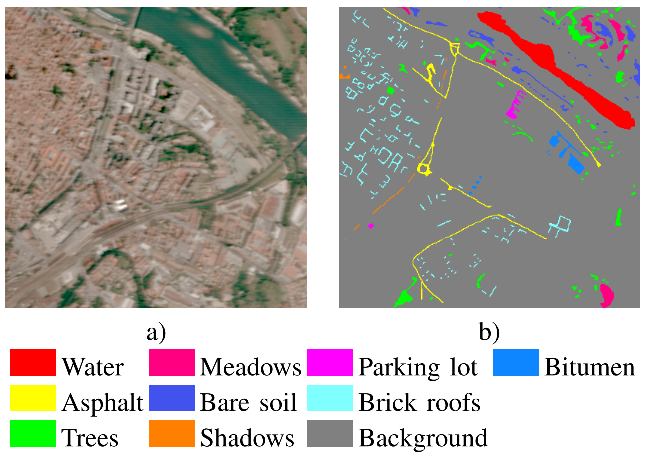


Fig. 12. a) False color Pavia hyperspectral image composed by bands [8, 3, 1], b) ground truth showing classes in colors and background in gray.

TABLE III
OVERALL ACCURACIES (%) HYPERSPECTRAL MULTICLASS IMAGE CLASSIFICATION PROBLEM

Realizations	GPC	LRF	NNF	SVMF	GPF
1	94.94	84.77	96.22	91.22	**96.50**
2	96.27	64.77	96.83	92.33	**97.61**
3	96.05	96.38	**96.44**	94.33	**96.44**
4	94.83	**97.11**	96.38	91.33	96.33
5	94.67	73.88	**97.22**	90.94	96.61
6	96.72	84.66	97.38	92.61	**97.61**
7	96.00	62.22	95.66	91.88	**97.27**
8	95.50	95.44	96.33	92.72	**97.33**
9	95.50	63.88	**97.11**	94.11	96.61
10	96.95	85.66	96.94	95.72	**97.05**
Mean	95.75	80.88	96.65	92.72	**96.94**

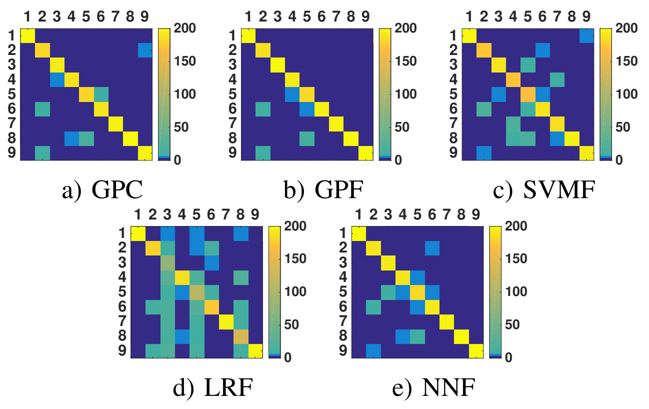
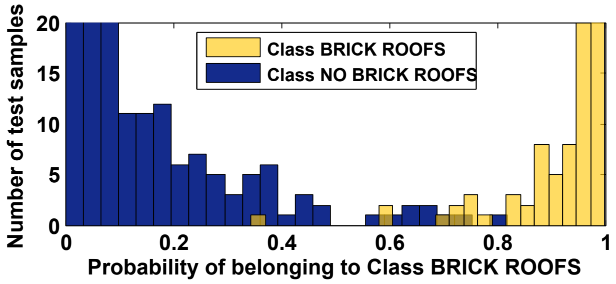
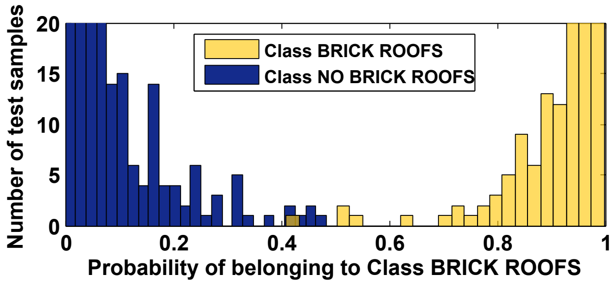


Fig. 13. Pavia hyperspectral image experiment: Mean Confusion Matrices for a) GPC, b) GPF, c) SVMF, d) LRF, e) NNF.



a)



b)

Fig. 14. Pavia hyperspectral image experiment: a) Histogram of probabilities of belonging to Class BRICK ROOFS with multiclass GPC, b) Histogram of probabilities of belonging to Class BRICK ROOFS with multiclass GPF.

To estimate all unknowns we have utilized variational inference and approximated the posterior distribution by minimizing the Kullback-Leibler divergence. This procedure has led to a parameter-free iterative algorithm.

In the experimental section, the proposed method has been compared for different scenarios, against a Gaussian process classifier built on non filtered features, convolutional neural networks and a filtering/classification approach based on the use of SVMs. In all cases the proposed method has outperformed all these methods in terms of accuracy. In addition, it is much more computationally efficient than its current SVM competitor.

References

[1] V. Mayer-Schnberger and K. Cukier, *Big Data: A Revolution That Will Transform How We Live, Work, and Think*, reprint edition ed. Boston: Eamon Dolan/Mariner Books, Mar. 2014.
[2] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, 1st ed. Cambridge, MA: The MIT Press, Aug. 2012.
[3] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data*. AMLBook, Mar. 2012.
[4] I. Amro, J. Mateos, M. Vega, R. Molina, and A. K. Katsaggelos, "A survey of classical methods and new trends in pansharpening of multispectral images," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, no. 79, September 2011.
[5] P. Ruiz, J. Mateos, G. Camps-Valls, R. Molina, and A. K. Katsaggelos, "Bayesian Active Remote Sensing Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 4, April 2014.
[6] B. Amizic, L. Spinoulas, R. Molina, and A. K. Katsaggelos, "Compressive sampling with unknown blurring function: application to passive millimiter-wave imaging," in *IEEE International Conference on Image Processing*. Orlando, Florida, October 2012, pp. 925–928.
[7] O. Martinez, L. Ferraz, X. Binefa, I. Gomez, and C. Dorronsoro, "Concealed object detection and segmentation over millimetric waves images," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2010, p. 3137.
[8] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, 1996.
[9] M. Elad, *Sparse and Redundant Representations - From Theory to Applications in Signal and Image Processing*. Springer, 2010.
[10] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ser. ICCV '99. IEEE Computer Society, 1999, pp. 1150–1157.
[11] R. Rigamonti, M. Brown, and V. Lepetit, "Are sparse representations really relevant for image classification?" in *Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1545–1552.
[12] R. Rigamonti, E. Türetken, G. G. Serrano, P. Fua, and V. Lepetit, "Filter learning for linear structure segmentation," École Polytechnique Fédérale de Lausanne (EPFL), Tech. Rep. 166951, 2011.
[13] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, pp. 145–175, 2001.
[14] N. D. Lawrence, "Gaussian process latent variable models for visualisation of high dimensional data," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004.
[15] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative learned dictionaries for local image analysis," in *CVPR*, 2008.
[16] I. Ramírez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features," in *CVPR*, 2010, pp. 3501–3508.
[17] D. Gutiérrez and D. I. D. I. Escalona-Vargas, "EEG data classification through signal spatial redistribution and optimized linear discriminants." *Computer Methods and Programs in Biomedicine*, vol. 97, no. 1, pp. 39–47, 2010.
[18] S. Ryali, K. Supekar, T. Chen, and V. Menon, "Multivariate dynamical systems models for estimating causal interactions in fMRI," *Journal of Neuroimmunology*, vol. 54, pp. 807–823, 2011.
[19] M. Varma and A. Zisserman, "A statistical approach to texture classification from single images," *International Journal of Computer Vision: Special Issue on Texture Analysis and Synthesis*, vol. 62, no. 1–2, pp. 61–81, April 2005.
[20] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2005, pp. 886–893.
[21] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2006, pp. 2169 –2178.
[22] Y. Lecun and Y. Bengio, *Convolutional Networks for Images, Speech and Time Series*. The MIT Press, 1995, pp. 255–258.
[23] B. de Vries and J. C. Príncipe, "The gamma model–a new neural model for temporal processing," *Neural Networks*, vol. 5, no. 4, pp. 565–576, 1992.
[24] R. Flamary, D. Tuia, B. Labbé, G. Camps-Valls, and A. Rakotomamonjy, "Large margin filtering," *IEEE Transactions Signal Processing*, vol. 60, no. 2, pp. 648–659, 2012.
[25] P. Ruiz, J. Mateos, R. Molina, and A. K. Katsaggelos, "Learning Filters in Gaussian Process Classification Problems," in *IEEE International Conference on Image Processing (ICIP)*, Paris (France), Oct. 2014.
[26] S. Boyd, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010. [Online]. Available: http://www.nowpublishers.com/article/Details/MAL-016
[27] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. NY: MIT Press, 2006.
[28] G. Camps-Valls and L. Bruzzone, *Kernel Methods for Remote Sensing Data Analysis*. John Wiley and Sons, 2009.
[29] G. Camps-Valls, D. Tuia, L. Gmez-Chova, S. Jimnez, and J. Malo, Eds., *Remote Sensing Image Processing*. LaPorte, CO, USA: Morgan & Claypool Publishers, Sep. 2011.
[30] L. Gomez-Chova, D. Fernández-Prieto, J. Calpe, E. Soria, J. Vila-Francés, and G. Camps-Valls, "Urban monitoring using multitemporal SAR and multispectral data," *Pattern Recognition Letters, Special Issue on "Pattern Recognition in Remote Sensing"*, vol. 27, no. 4, pp. 234–243, 2006.
[31] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.
[32] B. Amizic, L. Spinoulas, R. Molina, and A. K. Katsaggelos, "Variational bayesian compressive blind image deconvolution," in *21th European Signal Processing Conference (EUSIPCO 2013), Best Student Paper Award*. Marrakech (Morocco), September 2013.
[33] ——, "Compressive Blind Image Deconvolution," *IEEE Transactions on Image Processing*, vol. 22, no. 10, pp. 3994–4006, Oct. 2013.

[34] Z. Chen, S. Derin Babacan, R. Molina, and A. K. Katsaggelos, "Variational Bayesian Methods For Multimedia Problems," *IEEE Transactions on Multimedia*, vol. 16, no. 4, Jun. 2014.

[35] G. Parisi, *Statistical Field Theory*. Reading, Mass: Perseus Books, Nov. 1998.

[36] G. Bouchard, "Efficient bounds for the softmax function, applications to inference in hybrid models," in *2007 Neural Information Processing Systems Conference, NIPS 2007*, vol. 6239, 2007.

[37] S. Boyd and L. Vandenberghe, *Convex Optimization*, 1st ed. Cambridge, UK ; New York: Cambridge University Press, Mar. 2004.