1

A Review of Kernel Methods in Remote Sensing Data Analysis

Luis Gómez-Chova, Jordi Muñoz-Marí, Valero Laparra,
Jesús Malo-López and Gustavo Camps-Valls

Image Processing Laboratory, Universitat de València, Spain.
C/ Dr. Moliner, 50. Burjassot (València), Spain.
Tlf.: +34-963544024 - Fax: +34-963544353
luis.gomez-chova@uv.es, http://www.uv.es/chovago

Summary. Kernel methods have proven effective in the analysis of images of the Earth acquired by airborne and satellite sensors. Kernel methods provide a consistent and well-founded theoretical framework for developing nonlinear techniques and have useful properties when dealing with low number of (potentially high dimensional) training samples, the presence of heterogenous multimodalities, and different noise sources in the data. These properties are particularly appropriate for remote sensing data analysis. In fact, kernel methods have improved results of parametric linear methods and neural networks in applications such as natural resource control, detection and monitoring of anthropic infrastructures, agriculture inventorying, disaster prevention and damage assessment, anomaly and target detection, biophysical parameter estimation, band selection, and feature extraction.

This chapter provides a survey of applications and recent theoretical developments of kernel methods in the context of remote sensing data analysis. The specific methods developed in the fields of supervised classification, semisupervised classification, target detection, model inversion, and nonlinear feature extraction are revised both theoretically and through experimental (illustrative) examples. The emergent fields of transfer, active, and structured learning, along with efficient parallel implementations of kernel machines, are also revised.

1.1 Introduction

Remotely sensed images allow Earth Observation with unprecedented accuracy. New satellite sensors acquire images with high spectral and spatial resolution, and the revisiting time is constantly reduced. Processing data is becoming more complex in such situations and many problems can be tackled with recent machine learning tools. One of the most critical applications is that of image classification, but also model inversion and feature extraction are relevant in the field. This chapter will focus on these important problems that are subsequently outlined.

1.1.1 Classification with Kernels

The characteristics of the acquired images allow the characterization, identification, and classification of the land-covers [1]. However, traditional classifiers such as Gaussian maximum likelihood or artificial neural networks are affected by the high input sample dimension, tend to overfit data in the presence of noise, or perform poorly when a low number of training samples are available [2, 3]. In the last few years, the use of support vector machines (SVMs) [4, 5] for remote sensing image classification has been paid attention basically because the method integrates in the same classification procedure (i) a *feature extraction* step, as samples are mapped to a higher dimensional space where a simpler (linear) classification is performed, becoming nonlinear in the input space; (ii) a *regularization* procedure by which model's complexity is efficiently controlled; and (iii) the minimization of an upper bound of the generalization error, thus following the Structural Risk Minimization (SRM) principle. These theoretical properties, which will be reviewed in the next section, make the SVM in particular, and kernel methods in general, very attractive in the context of remote sensing image classification [6].

Another different concern is that a complete and representative training set is essential for a successful classification. In particular, it is noteworthy that few attention has been paid to the case of having an incomplete knowledge of the classes present in the investigated scene. This may be critical since, in many applications, acquiring ground truth information for all classes is very difficult, especially when complex and heterogeneous geographical areas are analyzed. In this chapter, we revise the one-class SVM for remotely-sensed image classification with incomplete training data. This method is a recent kernel-based development that only considers samples belonging to the class of interest in order to learn the underlying data class distribution. The method was originally introduced for anomaly detection [7], then analyzed for dealing with incomplete and unreliable training data [8], and recently reformulated for change detection [9].

Remote sensing image classification is hampered by both the number and quality of labeled training samples. In order to alleviate this problem, SVMs (or any other kernel-based classifier) should exploit the information contained in the abundant unlabeled samples along with the low number of labeled samples thus working under the semisupervised learning (SSL) paradigm [10]. In this chapter, we review the SSL literature and provide some experimental evidence of the use of semisupervised approaches for classification in challenging remote sensing problems.

1.1.2 Model Inversion with Kernels

Remote sensing very often deals with inverting a forward model. To this aim, one has to produce an accurate and robust model able to predict physical, chemical, geological or atmospheric parameters from spectra, such as surface temperature, water vapour, ozone, etc. This has been an active research field in remote sensing for years, and kernel methods offer promising non-parametric semi-empirical solutions. Kernel developments have been published in the last years: support vector regression (SVR) methods have been used for parameter estimation [11, 12, 13, 14], and a fully-constrained kernel least squares (FC-KLS) for abundance estimation [15]. Also, under a Bayesian perspective, other forms of kernel regression have been applied, such as the relevance vector machine (RVM) [16] or the Gaussian Process (GP) regression [17, 18].

1.1.3 Feature Extraction with Kernels

Recently, some attention has been paid to develop kernel-based feature extraction methods for remote sensing data processing. The main interest is to extract a reduced number of (nonlinear) features with high expressive power for either classification or regression. Particular applications to remote sensing are the Kernel Principal Component Analysis (KPCA) [5] and the Kernel Partial Least Squares (KPLS) [19].

The rest of this chapter is outlined as follows. Section 2 presents a brief introduction to kernel methods, fixes notation, and reviews the basic properties. Section 3 is devoted to review the classification setting, under the paradigms of supervised, semisupervised, and one-class classification. Section 4 presents the advances in kernel methods for regression and model inversion. Section 5 reviews the field of nonlinear feature extraction with kernels. Section 6 reviews the recent developments and foresees the future trends in kernel machines for remote sensing data analysis. Section 7 concludes the chapter with some final remarks.

1.2 Introduction to Kernel Methods

This section includes a brief introduction to kernel methods. After setting the scenario and fixing the most common notation, we give the main properties of kernel methods. We also pay attention to kernel methods development by means of particular properties drawn from linear algebra and functional analysis [20, 21].

1.2.1 Measuring Similarity with Kernels

Kernel methods rely on the notion of similarity between examples. Let us define a set of empirical data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathcal{X} \times \mathcal{Y}$, where $\mathbf{x}_i$ are the *inputs* taken from $\mathcal{X}$ and $y_i \in \mathcal{Y}$ are called the *outputs*. Learning means using these data pairs to predict well on test examples $\mathbf{x} \in \mathcal{X}$. To develop machines that generalize well, kernel methods try to exploit the structure of the data and thus define a similarity between pairs of samples.

Since $\mathcal{X}$ may not have a proper notion of similarity, examples are mapped to a (dot product) space $\mathcal{H}$, using a mapping $\boldsymbol{\phi} : \mathcal{X} \rightarrow \mathcal{H}, \mathbf{x} \mapsto \boldsymbol{\phi}(\mathbf{x})$. The similarity between the elements in $\mathcal{H}$ can now be measured using its associated dot product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$. Here, we define a function that computes that similarity, $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that $(\mathbf{x}, \mathbf{x}') \mapsto K(\mathbf{x}, \mathbf{x}')$. This function, called *kernel*, is required to satisfy:

$$K(\mathbf{x}, \mathbf{x}') = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle_{\mathcal{H}}. \tag{1.1}$$

The mapping $\boldsymbol{\phi}$ is its *feature map*, and the space $\mathcal{H}$ its *feature space*.

1.2.2 Positive Definite Kernels

The class of kernels that can be written in the form of (1.1) coincides with the class of positive definite kernels.

Definition 1. *A function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel* if and only if *there exists a Hilbert space $\mathcal{H}$ and a feature map $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{H}$ such that for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ we have $K(\mathbf{x}, \mathbf{x}') = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle_{\mathcal{H}}$.*

In practice, a real symmetric $n \times n$ matrix $\mathbf{K}$, whose entries are $K(\mathbf{x}_i, \mathbf{x}_j)$ or simply K_{ij}, is called *positive definite* if for all $c_1, \ldots, c_n \in \mathbb{R}$, $\sum_{i,j=1}^{n} c_i c_j K_{ij} \geq 0$. Note that a positive definite kernel is equivalent to a positive definite Gram matrix in the *feature space*.

Therefore, algorithms operating on the data only in terms of dot products can be used with any positive definite kernel by simply replacing $\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle_{\mathcal{H}}$ with kernel evaluations $K(\mathbf{x}, \mathbf{x}')$, a technique also known as the *kernel trick* [5]. Another direct consequence is that, for a positive definite kernel, one does not need to know the explicit form of the feature map since it is implicitly defined through the kernel.

1.2.3 Basic Operations with Kernels

We now review some basic properties with kernels. Note that, although the space $\mathcal{H}$ can be very high-dimensional, some basic operations can still be performed:

Translation. A translation in feature space can be written as the modified feature map $\tilde{\boldsymbol{\phi}}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x}) + \Gamma$ with $\Gamma \in \mathcal{H}$. Then, the translated dot product for $\langle \tilde{\boldsymbol{\phi}}(\mathbf{x}), \tilde{\boldsymbol{\phi}}(\mathbf{x}') \rangle_{\mathcal{H}}$ can be computed if we restrict Γ to lie in the span of the functions $\{\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_n)\} \in \mathcal{H}$.

Centering. The previous translation allows us to center data $\{\mathbf{x}_i\}_{i=1}^{n} \in \mathcal{X}$ in the *feature space*. The mean of the data in $\mathcal{H}$ is $\boldsymbol{\phi}_\mu = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\phi}(\mathbf{x}_i)$ which is a linear combination of the span of functions and thus fulfills the requirement for Γ. One can center data in $\mathcal{H}$ by computing $\mathbf{K} \leftarrow \mathbf{HKH}$ where entries of $\mathbf{H}$ are $H_{ij} = \delta_{ij} - \frac{1}{n}$ and the Kronecker symbol $\delta_{i,j} = 1$ if $i = j$ and zero otherwise.

Subspace Projections. Given two points Ψ and Γ in the feature space, the projection of Ψ onto the subspace spanned by Γ is $\Psi' = \frac{\langle \Gamma, \Psi \rangle_{\mathcal{H}}}{\|\Gamma\|_{\mathcal{H}}^2}\Gamma$. Therefore one can compute the projection Ψ' expressed solely in terms of kernel evaluations.

Computing Distances. The kernel corresponds to a dot product in a Hilbert Space $\mathcal{H}$, and thus one can compute distances between mapped samples entirely in terms of kernel evaluations:

$$d(\mathbf{x}, \mathbf{x}') = \|\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}')\|_{\mathcal{H}} = \sqrt{K(\mathbf{x}, \mathbf{x}) + K(\mathbf{x}', \mathbf{x}') - 2K(\mathbf{x}, \mathbf{x}')}$$

Normalization. Exploiting the previous property, one can also normalize data in feature spaces:

$$K(\mathbf{x}, \mathbf{x}') \leftarrow \left\langle \frac{\boldsymbol{\phi}(\mathbf{x})}{\|\boldsymbol{\phi}(\mathbf{x})\|}, \frac{\boldsymbol{\phi}(\mathbf{x}')}{\|\boldsymbol{\phi}(\mathbf{x}')\|} \right\rangle = \frac{K(\mathbf{x}, \mathbf{x}')}{\sqrt{K(\mathbf{x}, \mathbf{x})K(\mathbf{x}', \mathbf{x}')}}$$

1.2.4 Standard Kernels

The bottleneck for any kernel method is the definition of a kernel mapping function $\boldsymbol{\phi}$ that accurately reflects the similarity among samples. However, not all kernel similarity functions are permitted. In fact, valid kernels are only those fulfilling Mercer's Theorem (roughly speaking, being positive definite similarity matrices) and the most common ones are the linear $K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle$, the polynomial $K(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{z} \rangle + 1)^d$, $d \in \mathbb{Z}^+$, and the Radial Basis Function (RBF), $K(\mathbf{x}, \mathbf{z}) = \exp\left(-\|\mathbf{x} - \mathbf{z}\|^2/2\sigma^2\right)$, $\sigma \in \mathbb{R}^+$. Note that, by Taylor series expansion, the RBF kernel is a polynomial kernel with infinite degree. Thus the corresponding Hilbert space is infinite dimensional, which corresponds to a mapping into the space of smooth functions $\mathcal{C}^\infty$. The RBF kernel is also of practical convinience –stability and only one parameter to be tuned–, and it is the preferred kernel measure in standard applications.

1.2.5 Kernel Development

Taking advantage of some algebra and functional analysis properties [20, 21], one can derive very useful properties of kernels. Be K_1 and K_2 two positive definite kernels on $\mathcal{X} \times \mathcal{X}$, $\mathbf{A}$ a symmetric positive semidefinite matrix, $d(\cdot, \cdot)$ a metric fulfilling distance properties, f any function, and $\mu > 0$. Then, the following kernels are valid [5]:

$$K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') + K_2(\mathbf{x}, \mathbf{x}') \tag{1.2}$$
$$K(\mathbf{x}, \mathbf{x}') = \mu K_1(\mathbf{x}, \mathbf{x}') \tag{1.3}$$
$$K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') \cdot K_2(\mathbf{x}, \mathbf{x}') \tag{1.4}$$
$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{A} \mathbf{x}' \tag{1.5}$$
$$K(\mathbf{x}, \mathbf{x}') = \exp(-d(\mathbf{x}, \mathbf{x}')) \tag{1.6}$$
$$K(\mathbf{x}, \mathbf{x}') = K(f(\mathbf{x}), f(\mathbf{x}')) \tag{1.7}$$

These basic properties give rise to the construction of refined similarity measures better fitted to the data characteristics. In remote sensing, one can sum dedicated kernels to spectral, contextual or even temporal information of pixels through (1.2). A scaling factor to each kernel can also be added (Eq. 1.3).

Also, the (more appropriate) spectral angle distance between pixels is a valid kernel by (1.6). Recent advances for kernel development are:

Convex combinations. By exploiting (1.2) and (1.3), one can build kernels by linear combinations of kernels working on feature subsets:

$$K(\mathbf{x}, \mathbf{x}') = \sum_{m=1}^{M} d_m K_m(\mathbf{x}, \mathbf{x}')$$

This field of research is known as multiple kernel learning (MKL) and different algorithms exist to optimize the weights and kernel parameters jointly. Note that this kernel offers some insight in the problem, since relevant features receive higher values of d_m, and the corresponding kernel parameters θ_m yield information about pairwise similarity scales.

Deforming kernels. The field of semisupervised kernel learning deals with techniques to modify the values of the training kernel including the information from the whole data distribution: K is either deformed through a graph distance matrix built with both labeled and unlabeled samples, or by means of kernels built from clustering solutions.

Generative kernels. Exploiting Eq. (1.7), one can construct kernels from probability distributions by defining $K(\mathbf{x}, \mathbf{x}') = K(\mathbf{p}, \mathbf{p}')$, where $\mathbf{p}$, $\mathbf{p}'$ are defined on the space $\mathcal{X}$. This kind of kernels is known as *probability product kernels between distributions* and is defined as:

$$K(\mathbf{p}, \mathbf{p}') = \langle \mathbf{p}, \mathbf{p}' \rangle = \int_{\mathcal{X}} \mathbf{p}(\mathbf{x})\mathbf{p}'(\mathbf{x})\mathrm{d}\mathbf{x}.$$

Joint input-output mappings. Typically, kernels are built on the input samples. Lately the framework of *structured output learning* deals with the definition of joint input-output kernels, $K((\mathbf{x}, y), (\mathbf{x}', y'))$.

1.3 Kernel methods in remote sensing data classification

Classification maps are the main product of remote sensing data analysis and, in the last years, kernel methods have demonstrated very good performance. The most successful kernel method are the support vector machines as extensively reported in [6]. SVMs have been applied to both multispectral [22, 23] and hyperspectral [6, 24, 9] data in a wide range of domains, including object recognition [25], land cover and multi-temporal classification [26, 27, 9], and urban monitoring [28].

1.3.1 Support Vector Machine (SVM)

The Support Vector Machine (SVM) is defined as follows. Notationally, given a labeled training data set $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$, where $\mathbf{x}_i \in \mathbb{R}^N$ and $y_i \in \{-1, +1\}$, and given a nonlinear mapping $\boldsymbol{\phi}(\cdot)$, the SVM method solves:

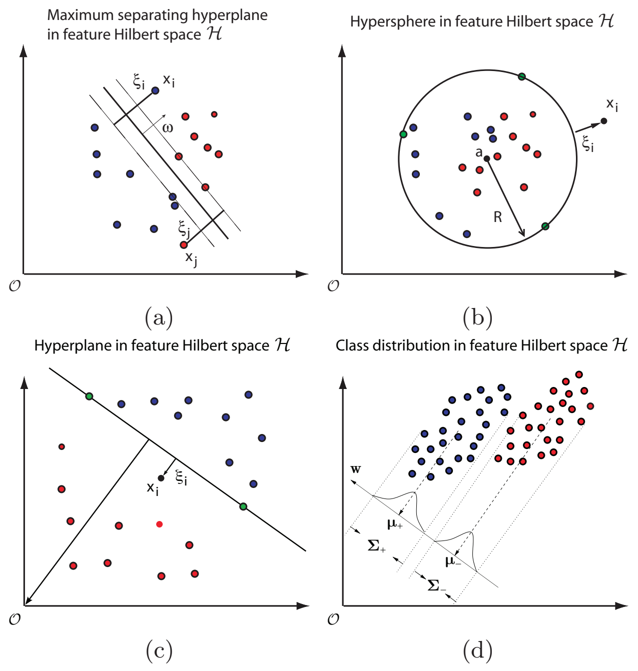
Maximum separating hyperplane in feature Hilbert space $\mathcal{H}$
ξ_i
x_i
ω
ξ_j
x_j
$\mathcal{O}$
(a)
Hypersphere in feature Hilbert space $\mathcal{H}$
x_i
ξ_i
a
R
$\mathcal{O}$
(b)
Hyperplane in feature Hilbert space $\mathcal{H}$
x_i
ξ_i
$\mathcal{O}$
(c)
Class distribution in feature Hilbert space $\mathcal{H}$
w
μ_+
Σ_+
μ_-
Σ_-
$\mathcal{O}$
(d)

Fig. 1.1. Illustration of kernel classifiers. (a) SVM: Linear decision hyperplanes in a nonlinearly transformed, feature space, where *slack* variables ξ_i are included to deal with errors. (b) SVDD: The hypersphere containing the target data is described by center $\mathbf{a}$ and radius R. Samples in the boundary and outside the ball are unbounded and bounded support vectors, respectively. (c) OC-SVM: another way of solving the data description problem, where all samples from the target class are mapped with maximum distance to the origin. (d) KFD: Kernel Fisher's Discriminant separates the classes by projecting them onto a hyperplane where the difference of the projected means (μ_1, μ_2) is large, and the variance around means σ_1 and σ_2 is small.

$$\min_{\mathbf{w},\xi_i,b}\left\{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i\right\} \tag{1.8}$$

constrained to:

$$y_i(\langle\boldsymbol{\phi}(\mathbf{x}_i),\mathbf{w}\rangle + b) \geq 1 - \xi_i \qquad \forall i = 1,\ldots,n \tag{1.9}$$

$$\xi_i \geq 0 \qquad \forall i = 1,\ldots,n \tag{1.10}$$

where $\mathbf{w}$ and b define a linear classifier in the feature space, and ξ_i are positive slack variables enabling to deal with permitted errors (Fig. 1.1a). Appropriate choice of nonlinear mapping $\boldsymbol{\phi}$ guarantees that the transformed samples are more likely to be linearly separable in the (higher dimension) feature space. The regularization parameter C controls the generalization capability of the classifier, and it must be selected by the user. Primal problem (1.8) is solved using its dual problem counterpart [5], and the decision function for any test vector $\mathbf{x}_*$ is finally given by

$$f(\mathbf{x}_*) = sgn\left(\sum_{i=1}^{n} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}_*) + b\right) \tag{1.11}$$

where α_i are Lagrange multipliers corresponding to constraints in (1.9), being the support vectors (SVs) those training samples $\mathbf{x}_i$ with non-zero Lagrange multipliers $\alpha_i \neq 0$; $K(\mathbf{x}_i, \mathbf{x}_*)$ is an element of a kernel matrix $\mathbf{K}$ defined as in equation (1.1); and the bias term b is calculated by using the *unbounded* Lagrange multipliers as $b = 1/k \sum_{i=1}^{k}(y_i - \langle \boldsymbol{\phi}(\mathbf{x}_i), \mathbf{w}\rangle)$, where k is the number of *unbounded* Lagrange multipliers ($0 \leqslant \alpha_i < C$) and $\mathbf{w} = \sum_{i=1}^{n} y_i \alpha_i \boldsymbol{\phi}(\mathbf{x}_i)$ [5].

1.3.2 ν-Support Vector Machine (ν-SVM)

One interesting variation of the SVM is the ν-SVM introduced by Schölkopf et al [29]. In the SVM formulation, the soft margin is controlled by parameter C, which may take any positive value. This makes difficult to adjust it when training the classifier. The idea of the ν-SVM is forcing the soft margin to lie in the range $[0, 1]$. This is carried out redefining the problem as

$$\min_{\mathbf{w},\xi_i,b,\rho} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \nu\rho + \frac{1}{n}\sum_{i=1}^{n} \xi_i \right\} \tag{1.12}$$

subject to:

$$y_i(\langle \boldsymbol{\phi}(\mathbf{x}_i), \mathbf{w}\rangle + b) \geq \rho - \xi_i \qquad \forall i = 1, \ldots, n \tag{1.13}$$

$$\rho \geq 0, \xi_i \geq 0 \qquad \forall i = 1, \ldots, n \tag{1.14}$$

In this new formulation, parameter C has been removed and a new variable ρ with coefficient ν has been introduced. This new variable ρ adds another degree of freedom to the margin, the size of the margin increasing linearly with ρ. The old parameter C controlled the trade off between the training error and the generalization error. In the ν-SVM formulation, this is done adjusting ν in the range $[0, 1]$, which acts as an upper bound on the fraction of margin errors, and it is also a lower bound on the fraction of support vectors.

1.3.3 Support Vector Data Description (SVDD)

A different problem statement for classification is given by the Support Vector Domain Description (SVDD) [30]. The SVDD is a method to solve one-class problems, where one tries to describe one class of objects, distinguishing them from all other possible objects.

The problem is defined as follows. Let $\{\mathbf{x}_i\}_{i=1}^{n}$ be a dataset belonging to a given *class of interest*. The purpose is to find a minimum volume *hypersphere* in a high dimensional feature space $\mathcal{H}$, with radius $R > 0$ and center $\mathbf{a} \in \mathcal{H}$, which contains most of these data objects (Fig. 1.1b). Since the training set may contain outliers, a set of *slack variables* $\xi_i \geq 0$ is introduced, and the problem becomes

$$\min_{R,\mathbf{a}} \left\{ R^2 + C \sum_{i=1}^{n} \xi_i \right\} \tag{1.15}$$

constrained to

$$\|\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{a}\|^2 \leq R^2 + \xi_i \qquad \forall i = 1, \ldots, n \tag{1.16}$$

$$\xi_i \geq 0 \qquad \forall i = 1, \ldots, n \tag{1.17}$$

where parameter C controls the trade-off between the volume of the hypersphere and the permitted errors. Parameter ν, defined as $\nu = 1/nC$, can be used as a rejection fraction parameter to be tuned as noted in [31].

The dual functional is a quadratic programming problem that yields a set of Lagrange multipliers (α_i) corresponding to constraints in (1.16). When the free parameter C is adjusted properly, most of the α_i are zero, giving a sparse solution. The Lagrange multipliers are also used to calculate the distance from a test point to the center $R(\mathbf{x}_*)$:

$$R(\mathbf{x}_*) = K(\mathbf{x}_*, \mathbf{x}_*) - 2 \sum_{i=1}^{n} K(\mathbf{x}_i, \mathbf{x}_*) + \sum_{i,j=1}^{n} K(\mathbf{x}_i, \mathbf{x}_j) \tag{1.18}$$

which is compared with ratio R. Unbounded support vectors are those samples $\mathbf{x}_i$ satisfying $0 \leqslant \alpha_i < C$, while bounded SVs are samples whose associated $\alpha_i = C$, which are considered outliers.

1.3.4 One-class Support Vector Machine (OC-SVM)

In the OC-SVM, instead of defining a hypersphere containing all examples, a hyperplane that separates the data objects from the origin with maximum margin is defined (Fig. 1.1c). It can be shown that when working with normalized data and the RBF Gaussian kernel, both methods yield the same solutions [31].

In the OC-SVM, we want to find a hyperplane $\mathbf{w}$ which separates samples $\mathbf{x}_i$ from the origin with margin ρ. The problem thus becomes

$$\min_{\mathbf{w},\rho,\xi} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 - \rho + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i \right\} \tag{1.19}$$

constrained to

$$\langle \boldsymbol{\phi}(\mathbf{x}_i), \mathbf{w} \rangle \geq \rho - \xi_i \qquad \forall i = 1, \ldots, n \tag{1.20}$$

$$\xi_i \geq 0 \qquad \forall i = 1, \ldots, n \tag{1.21}$$

The problem is solved through its Langrangian dual introducing a set of Lagrange multipliers α_i. The margin ρ can be computed as $\rho = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle = \sum_j \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$.

1.3.5 Kernel Fisher's Discriminant (KFD)

Assume that, n_1 out of n training samples belong to class -1 and n_2 to class $+1$. Let $\boldsymbol{\mu}$ be the mean of the whole set, and $\boldsymbol{\mu}_-$ and $\boldsymbol{\mu}_+$ the means for classes -1 and $+1$, respectively. Analogously, let $\boldsymbol{\Sigma}$ be the covariance matrix of the whole set, and $\boldsymbol{\Sigma}_-$ and $\boldsymbol{\Sigma}_+$ the covariance matrices for the two classes.

The Linear Fisher's Discriminant (LFD) seeks for projections that maximize the interclass variance and minimize the intraclass variance [32, 33]. By defining the *between class scatter matrix* $\mathbf{S}_B = (\boldsymbol{\mu}_- - \boldsymbol{\mu}_+)(\boldsymbol{\mu}_- - \boldsymbol{\mu}_+)^\top$ and the *within class scatter matrix* $\mathbf{S}_W = \boldsymbol{\Sigma}_- + \boldsymbol{\Sigma}_+$, the problem reduces to maximize

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}} \tag{1.22}$$

The Kernel Fisher's Discriminant (KFD) is obtained by defining the LFD in a high dimensional *feature* space $\mathcal{H}$. Now, the problem reduces to maximize:

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top \mathbf{S}_B^{\boldsymbol{\phi}} \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W^{\boldsymbol{\phi}} \mathbf{w}} \tag{1.23}$$

where now $\mathbf{w}$, $\mathbf{S}_B^{\boldsymbol{\phi}}$ and $\mathbf{S}_W^{\boldsymbol{\phi}}$ are defined in $\mathcal{H}$, $\mathbf{S}_B^{\boldsymbol{\phi}} = (\boldsymbol{\mu}_-^{\boldsymbol{\phi}} - \boldsymbol{\mu}_+^{\boldsymbol{\phi}})(\boldsymbol{\mu}_-^{\boldsymbol{\phi}} - \boldsymbol{\mu}_+^{\boldsymbol{\phi}})^\top$, and $\mathbf{S}_W^{\boldsymbol{\phi}} = \boldsymbol{\Sigma}_-^{\boldsymbol{\phi}} + \boldsymbol{\Sigma}_+^{\boldsymbol{\phi}}$.

We need to express (1.23) in terms of dot-products only. According to the reproducing kernel theorem [5], any solution $\mathbf{w} \in \mathcal{H}$ can be represented as a linear combination of training samples in $\mathcal{H}$. Therefore $\mathbf{w} = \sum_{i=1}^n \alpha_i \boldsymbol{\phi}(\mathbf{x}_i)$ and then

$$\mathbf{w}^\top \boldsymbol{\mu}_i^{\boldsymbol{\phi}} = \frac{1}{n_i} \sum_{j=1}^{n} \sum_{k=1}^{n_i} \alpha_j K(\mathbf{x}_j, \mathbf{x}_k^i) = \boldsymbol{\alpha}^\top \mathbf{M}_i \tag{1.24}$$

where $\mathbf{x}_k^i$ represents samples $\mathbf{x}_k$ of class i, and $(\mathbf{M}_i)_j = \frac{1}{n_i} \sum_{k=1}^{n_i} K(\mathbf{x}_j, \mathbf{x}_k^i)$. Taking the definition of $\mathbf{S}_B^{\boldsymbol{\phi}}$ and (1.24), the numerator of (1.23) can be rewritten as $\mathbf{w}^\top \mathbf{S}_B^{\boldsymbol{\phi}} \mathbf{w} = \boldsymbol{\alpha}^\top \mathbf{M} \boldsymbol{\alpha}$, and the denominator as $\mathbf{w}^\top \mathbf{S}_W^{\boldsymbol{\phi}} \mathbf{w} = \boldsymbol{\alpha}^\top \mathbf{N} \boldsymbol{\alpha}$, where

$$\mathbf{M} = (\mathbf{M}_- - \mathbf{M}_+)(\mathbf{M}_- - \mathbf{M}_+)^\top \tag{1.25}$$

$$\mathbf{N} = \textstyle\sum_{j=\{-1,+1\}} \mathbf{K}_j (\mathbf{I} - \mathbf{1}_{n_j}) \mathbf{K}_j^\top \tag{1.26}$$

$\mathbf{K}_j$ is a $n \times n_j$ matrix with $(\mathbf{K}_j)_{nm} = K(\mathbf{x}_n, \mathbf{x}_m^j)$ (the kernel matrix for class j), $\mathbf{I}$ is the identity and $\mathbf{1}_{n_j}$ a matrix with all entries set to $1/n_j$. Finally, Fisher's linear discriminant in $\mathcal{H}$ is solved by maximizing

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^\top \mathbf{M} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top \mathbf{N} \boldsymbol{\alpha}}, \tag{1.27}$$

which is solved as in the linear case. The projection of a new sample $\mathbf{x}$ onto $\mathbf{w}$ can be computed through the kernel function:

$$\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \tag{1.28}$$

Table 1.1. Mean and standard deviation of estimated kappa statistic (κ), precision, recall, F-Measure and rate of support vectors for the 10 realizations. Best results are boldfaced.

Method	κ	Precicision	Recall	F-Measure	% SVs
ν-SVC lin	0.81 ± 0.06	0.83 ± 0.07	$\mathbf{0.90 \pm 0.07}$	0.86 ± 0.04	33 ± 0.13
ν-SVC RBF	0.80 ± 0.07	0.86 ± 0.08	0.85 ± 0.10	0.85 ± 0.05	36 ± 0.24
LFD	0.72 ± 0.06	0.76 ± 0.08	0.84 ± 0.05	0.79 ± 0.04	-
KFD	$\mathbf{0.82 \pm 0.03}$	0.87 ± 0.04	0.86 ± 0.05	$\mathbf{0.86 \pm 0.02}$	-
OC-SVM lin	0.70 ± 0.06	0.78 ± 0.11	0.79 ± 0.13	0.77 ± 0.05	15 ± 0.05
OC-SVM RBF	0.68 ± 0.16	$\mathbf{0.93 \pm 0.06}$	0.64 ± 0.21	0.74 ± 0.15	37 ± 0.12

1.3.6 Experimental Results for Supervised Classification

Here we compare the performance of ν-SVM, OC-SVM, LFD and KFD methods in a remote sensing multisource image classication problem: the identification of classes ‘urban’ and ‘non-urban’. For the ν-SVM, LFD and KFD the problem is binary. For OC-SVM, we take the class ‘urban’ as the target class. The images used are from ERS2 SAR and Landsat TM sensors acquired in 1999 over the area of Naples, Italy [34]. The dataset have seven Landsat bands, two SAR backscattering intensities (0–35 days), and the SAR interferometric coherence. Since these features come from different sensors, the first step was to perform a specific processing and conditioning of optical and SAR data, and to co-register all images. Then, all features were stacked at a pixel level. A small area of the image of 400×400 pixels was selected.

We used 10 randomly selected samples of each class to train the classifiers (only 10 ‘urban’ samples for the one-class experiment). Except the LFD, the other classifiers have free parameters that must be tuned in the training process. To do this, the training set was split following a v-fold strategy[1]. For all methods, we used the RBF kernel where σ was tuned in the range $[10^{-3}, 10^{3}]$ in logarithmic increments of 10. The ν-SVM and OC-SVM have and additional parameter to tune: ν was varied in the range $[0.1, 0.5]$ in increments of 0.1. Experiments were repeated 10 times with different random realizations of the training sets. Averaged results are shown using four different error measures obtained from the confusion matrices: the estimated kappa statistic (κ) [35]; the *precision* (P), defined as the ratio between the number of true positives and the sum of true positives and false positives; the *recall* (R), defined as the ratio between the number of true positives and the sum of true positives and false negatives. The last one is the F-Measure (or unbiased F-Score), computed as $F = 2\frac{P \cdot R}{P+R}$, which combines both measures. Table 1.1 shows the mean results and the percentage of support vectors for the 10 different training sets.

[1] In v-fold, the training set is divided in v subsets, then during v times $v-1$ subsets are used for training, and the remaining subset is used for validation. At the end, the parameters that have worked the best in the v subsets are selected.

Linear vs. nonlinear. From Table 1.1, several conclusions can be obtained concerning the suitable kernel. In the case of ν-SVM, linear kernel yields slightly favourable results but differences to the RBF kernel are not statistically significant. On the contrary, for the case of Fisher's discriminants, KFD is better than the linear kernel LFD. Particularly interesting is the case of the OC-SVM. Here, using the RBF Gaussian kernel has the problem of adjusting the width σ using only samples from the target class. The problem is quite difficult because, as reliable measures like the estimated kappa statistic or the F-Measure cannot be computed using only samples of the target class, σ should be adjusted by measuring only the true positive ratio and controlling model's complexity through the rate of support vectors. In those cases where a proper value for σ cannot be found, the linear kernel may perform better, as it has no free parameter to adjust.

ν-SVM vs. OC-SVM. In terms of the estimated kappa statistic, the ν-SVM classifier generally works better than the OC-SVM in this example. This result is not surprising since this experiment is essentially a binary problem and the ν-SVM has, in the training phase, information about both classes, whereas the OC-SVM is trained using only information of the class 'urban'. Comparing the results in terms of precision, the ν-SVM performs better than OC-SVM using the linear kernel, but worse when OC-SVM uses the RBF kernel. On the other hand, the ν-SVM obtains better results in terms of recall, meaning that it has less false negatives for the target class. Evaluating the performance with the F-Measure, which takes into account both precision and recall, the ν-SVM obtains better overall results. Finally, results clearly show that sparser classifiers are obtained when using the OC-SVM with the linear kernel.

Support Vector vs. Fisher's Discriminant. Algorithms based on support vectors using the RBF kernel have a similar (but slightly lower) performance than the KFD algorithm. This better performance may be due to the low number of training samples used (being non-sparse, KFD has a full –dense– representation of the training data) and the squared loss function used is better suited to the assumed (Gaussian) noise in the data.

1.3.7 Semisupervised Image Classification

Remote sensing image classification is a challenging task because only a small number of labeled pixels is typically available, and thus classifiers tend to overfit the data [2]. In this context, semisupervised learning (SSL) naturally appears as a promising tool for combining labeled and unlabeled information thus increasing the accuracy and robustness of class predictions [10, 36]. The key issue in SSL is the general assumption of *consistency*, which means that: 1) nearby points are likely to have the same label; and 2) points on the same data structure (cluster or manifold) are likely to have the same label. This argument is often called the *cluster assumption* [37, 38]. Traditional SSL methods are based on generative models, which estimate the conditional density and have been extensively applied in remote sensing image classification [39]. Recently, more attention has been paid to *discriminative* approaches, such as: 1) the

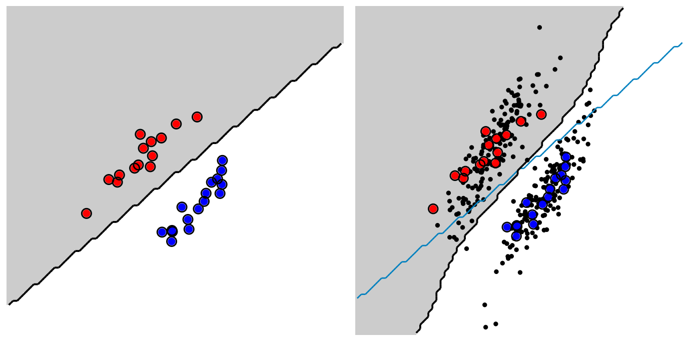

Fig. 1.2. Left: classifier obtained using labeled data (red and blue denote different classes). Right: classifier obtained using labeled data plus unlabeled data distribution (black dots denote unlabeled data).

Transductive SVM (TSVM) [4,40], which maximizes the margin for labeled and unlabeled samples simultaneously; 2) Graph-based methods, in which each pixel spreads its label information to its neighbors until a global steady state is achieved on the whole image [41,42]; and 3) the Laplacian SVM (LapSVM) [43,44], which deforms the kernel matrix of a standard SVM with the relations found by building the graph Laplacian. Also, the design of cluster and bagged kernels [37] have been successfully presented in remote sensing [45,46], whose essential idea is to modify the eigenspectrum of the kernel matrix that in turn implies an alteration of the distance metric. Figure 1.2 illustrates a typical semisupervised learning situation where distribution of unlabeled samples helps to improve the generalization of the classifier.

Manifold-based Regularization Framework

Regularization helps to produce smooth decision functions that avoid overfitting to the training data. Since the work of Tikhonov [47], many regularized algorithms have been proposed to control the capacity of the classifier [48,5]. Regularization has been applied to both linear and nonlinear algorithms in the context of remote sensing image classification, and becomes strictly necessary when few labeled samples are available compared to the high dimensionality of the problem. In the last decade, the most paradigmatic case of regularized nonlinear algorithm is the support vector machine: in this case, maximizing the margin is equivalent to applying a kind of regularization to model weights [5,6]. These regularization methods are especially appropriate when a low number of samples is available, but are not concerned about the geometry of the marginal data distribution. This has been recently treated within a more general regularization framework that includes Tikhonov's as a special case.

Semisupervised Regularization Framework

The classical regularization framework has been recently extended to the use of unlabeled samples [43] as follows. Notationally, we are given a set of l labeled samples, $\{\mathbf{x}_i\}_{i=1}^{l}$ with corresponding class labels y_i, and a set of u unlabeled samples $\{\mathbf{x}_i\}_{i=l+1}^{l+u}$. Let us now assume a general-purpose decision function f. The regularized functional to minimize is:

$$\mathcal{L} = \frac{1}{l}\sum_{i=1}^{l} V(\mathbf{x}_i, y_i, f) + \gamma_L \|f\|_{\mathcal{H}}^2 + \gamma_M \|f\|_{\mathcal{M}}^2, \tag{1.29}$$

where V represents a generic cost function of the committed errors on the labeled samples, γ_L controls the complexity of f in the associated Hilbert space $\mathcal{H}$, and γ_M controls its complexity in the intrinsic geometry of the data distribution. For example, if the probability distribution is supported on a low-dimensional manifold, $\|f\|_{\mathcal{M}}^2$ penalizes f along that manifold $\mathcal{M}$. Note that this semisupervised learning framework allows us to develop many different algorithms just by playing around with the loss function, V, and the regularizers, $\|f\|_{\mathcal{H}}^2$ and $\|f\|_{\mathcal{M}}^2$.

Laplacian Support Vector Machine (LapSVM)

Here, we briefly review the Laplacian SVM as an instantiation of the previous framework. More details can be found in [43], and its application to remote sensing data classification in [44].

The LapSVM uses the same hinge loss function as the traditional SVM:

$$V(\mathbf{x}_i, y_i, f) = \max(0, 1 - y_i f(\mathbf{x}_i)), \tag{1.30}$$

where f represents the decision function implemented by the selected classifier and the predicted labels are $y_* = \operatorname{sgn}(f(\mathbf{x}_*))$. Hereafter, unlabeled or test samples are highlighted with $*$.

The decision function used by the LapSVM is $f(\mathbf{x}_*) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_*)\rangle + b$, where $\boldsymbol{\phi}(\cdot)$ is a nonlinear mapping to a higher dimensional Hilbert space $\mathcal{H}$, and $\mathbf{w}$ and b define a linear decision function in that space. The decision function is given by $f(\mathbf{x}_*) = \sum_{i=1}^{l+u} \alpha_i K(\mathbf{x}_i, \mathbf{x}_*) + b$. The regularization term $\|f\|_{\mathcal{H}}^2$ can be fully expressed in terms of the corresponding kernel matrix and the expansion coefficients $\boldsymbol{\alpha}$:

$$\|f\|_{\mathcal{H}}^2 = \|\mathbf{w}\|^2 = (\boldsymbol{\Phi}\boldsymbol{\alpha})^\top(\boldsymbol{\Phi}\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}. \tag{1.31}$$

Essentially, for manifold regularization, the LapSVM relies on the Laplacian eigenmaps (LE), which tries to map nearby inputs (pixels) to nearby outputs (corresponding class labels), thus preserving the neighborhood relations between samples[2]. Therefore, the geometry of the data is modeled with a

[2] In our case, nearby points are those pixels spectrally similar and thus the assumption is applied to the (high) dimensional space of image pixels.

graph in which nodes represent both labeled and unlabeled samples connected by weights W_{ij} [10]. Regularizing the graph follows from the *smoothness* (or *manifold*) assumption and intuitively is equivalent to penalize "rapid changes" of the classification function evaluated between close samples in the graph:

$$\|f\|_{\mathcal{M}}^2 = \frac{1}{(l+u)^2} \sum_{i,j=1}^{l+u} W_{ij}(f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 = \frac{\mathbf{f}^\top \mathbf{L}\mathbf{f}}{(l+u)^2}, \qquad (1.32)$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the graph Laplacian, whose entries are sample and graph-dependent; $\mathbf{D}$ is the diagonal degree matrix of $\mathbf{W}$ given by $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$ and $D_{ij} = 0$ for $i \neq j$; the normalizing coefficient $\frac{1}{(l+u)^2}$ is the natural scale factor for the empirical estimate of the Laplace operator [43]; and $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{l+u})]^\top = \mathbf{K}\boldsymbol{\alpha}$, where we deliberately dropped the bias term b.

Transductive SVM (TSVM)

The TSVM, originally proposed in [4] and further extended to deal with the peculiarities of remote sensing data in [40], aims at choosing a decision boundary that maximizes the margin on both labeled and unlabeled data. The TSVM optimizes a loss function similar to (1.29), but $\gamma_M \|f\|_{\mathcal{M}}^2$ is replaced by a term related to the distance of unlabeled samples to the margin. The TSVM functional to be minimized is:

$$\mathcal{L} = \frac{1}{l} \sum_{i=1}^{l} V(\mathbf{x}_i, y_i, f) + \gamma_L \|f\|_{\mathcal{H}}^2 + \lambda \sum_{j=l+1}^{l+u} L^*(f(\mathbf{x}_j^*)), \qquad (1.33)$$

where l and u are the number of labeled and unlabeled examples, λ is a free parameter that controls the relevance of unlabeled samples, and L^* is the symmetric hinge loss function:

$$L^*(f(\mathbf{x}^*)) = \max(0, 1 - |f(\mathbf{x}^*)|). \qquad (1.34)$$

The optimization of L^* can be seen as "self-learning", i.e., we use the prediction for $\mathbf{x}^*$ for training the mapping for that same example. Minimizing (1.34) pushes away unlabeled samples from the margin, either negative or positive, thus minimizes the absolute value.

1.3.8 Experimental Results for Semisupervised Classification

This section presents the experimental results of semisupervised methods in the same urban monitoring application presented in the previous section [34]. However, different sets of labeled and unlabeled training samples were used in order to test the performance of the SSL methods. Training and validation sets consisting of $l = 400$ labeled samples (200 samples *per* class) were generated,

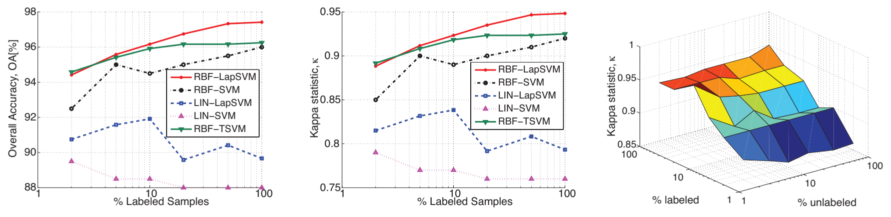

Fig. 1.3. Results for the urban classification. Overall Accuracy OA[%] (*left*) and Kappa statistic κ (*middle*) over the validation set as a function of the rate of labeled training samples used to build models. Kappa statistic surface (*right*) over the validation set for the best RBF-LapSVM classifier as a function of the rate of both labeled and unlabeled training samples.

and $u = 400$ unlabeled (randomly selected) samples from the analyzed images were added to the training set for the LapSVM and TSVM. We focus on the ill-posed scenario and vary the rate of both labeled and unlabeled samples independently, i.e. $\{2, 5, 10, 20, 50, 100\}\%$ of the labeled/unlabeled samples of the training set were used to train the models in each experiment. In order to avoid skewed conclusions, we run all experiments for a number of realizations where the used training samples were randomly selected.

Both linear and RBF kernels were used in the SVM, LapSVM, and TSVM. The graph Laplacian, $\mathbf{L}$, consisted of $l + u$ nodes connected using k nearest neighbors, and computed the edge weights W_{ij} using the Euclidean distance among samples. Free parameters γ_L and γ_M were varied in steps of one decade in the range $[10^{-4}, 10^4]$, the number of neighbors k used to compute the graph Laplacian was varied from 3 to 9, and the Gaussian width was tuned in the range $\sigma = \{10^{-3}, \ldots, 10\}$ for the RBF kernel. The selection of the best subset of free parameters was done by cross-validation.

Fig. 1.3 shows the validation results for the analyzed SVM-based classifiers. Several conclusions can be obtained. First, LapSVM classifiers produce better classification results than SVM in all cases (note that SVM is a particular case of the LapSVM for $\gamma_M = 0$) for both the linear and the RBF kernels. LapSVM also produces better classification results than TSVM when the number of labeled samples is increased. Differences among methods are numerically very similar when a low number of labeled samples is available. The κ surface for the LapSVM highlights the importance of the labeled information in this problem.

1.4 Kernel methods in biophysical parameter estimation

Robust, fast and accurate regression tools are a critical demand in remote sensing. The estimation of physical parameters, $\mathbf{y}$, from raw measurements, $\mathbf{x}$, is of special relevance in order to better understand the environment dynamics at local and global scales [49]. The inversion of analytical models introduces a higher level of complexity, induces an important computational burden, and sensitivity to noise becomes an important issue. In the recent years, nevertheless, the use of *empirical models* adjusted to learn the relationship between the acquired spectra and actual ground measurements has become very attractive. *Parametric* models have some important drawbacks, which typically lead to poor prediction results on unseen (test) data. As a consequence, *non-parametric* and potentially *nonlinear* regression techniques have been effectively introduced [50]. Different models and architectures of neural networks have been considered for the estimation of biophysical parameters [50,51,52]. However, despite their potential effectiveness, neural networks present some important drawbacks: (i) design and training often results in a complex, time-consuming task; (ii) following the minimization of the empirical risk (i.e. the error in the training data set), rather than the structural risk (an upper bound of the generalization error), can lead to overfit the training data; and (iii) performance can be degraded when working with low-sized data sets. A promising alternative to neural networks is the use of kernel methods analyzed in this section, such as support vector regression (SVR) [11,53], relevance vector machines (RVM) [16], and Gaussian Processes (GP) [17].

1.4.1 Support Vector Regression (SVR)

The support vector regression (SVR) is the SVM implementation for regression and function approximation [5, 54], which has yielded good results in modeling some biophysical parameters and in alleviating the aforementioned problems of neural networks [55, 56, 11].

The standard SVR formulation uses Vapnik's ε-insensitive cost function, in which errors e_i up to ε are not penalized, and all further deviations will incur in a linear penalization. Briefly, SVR estimates weights $\mathbf{w}$ by minimizing the following regularized functional:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i(\xi_i + \xi_i^*) \tag{1.35}$$

with respect to $\mathbf{w}$ and $\{\xi_i^{(*)}\}_{i=1}^n$, constrained to:

$$y_i - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) - b \leq \varepsilon + \xi_i \qquad \forall i = 1,\ldots,n \tag{1.36}$$

$$\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) + b - y_i \leq \varepsilon + \xi_i^* \qquad \forall i = 1,\ldots,n \tag{1.37}$$

$$\xi_i, \xi_i^* \geq 0 \qquad \forall i = 1,\ldots,n \tag{1.38}$$

where $\xi_i^{(*)}$ are positive slack variables to deal with training samples with a prediction error larger than ε ($\varepsilon > 0$), and C is the penalization parameter applied to these ones. Note that C trade-offs the minimization of errors and the regularization term, thus controling the generalization capabilities. The usual procedure for solving SVRs introduces the linear restrictions (1.36)-(1.38) into (1.35) using Lagrange multipliers α_i, computes the Karush-Kuhn-Tucker conditions, and solves the dual problem using QP procedures [57], which yields the final solution:

$$\hat{y}_i = \sum_{j=1}^{n} (\alpha_j - \alpha_j^*) K(\mathbf{x}_i, \mathbf{x}_j) + b. \tag{1.39}$$

Again, non-zero multipliers are called SVs. Sparsity in the SVR is a direct consequence of the loss function; as the value of ε increases, the number of support vectors is reduced.

1.4.2 Relevance Vector Machines (RVM)

Despite the good performance offered by the SVR, it has some deficiencies: (i) by assuming an explicit loss function (usually, the ε-insensitive loss function) one assumes a fixed distribution of the residuals, (ii) the free parameters must be tuned usually through cross-validation methods, which result in time consuming tasks, (iii) the nonlinear function used in SVR must fulfil Mercer's Theorem [58] to be valid, and (iv) sparsity is not always achieved and a high number of support vectors is thus obtained.

Some of these problems of SVRs are efficiently alleviated by the Relevance Vector Machine (RVM), which was originally introduced by Tipping in [59]. The RVM constitutes a Bayesian approximation to solve extended linear (in the parameters) models, i.e. nonlinear models. Therefore, the RVM follows a different inference principle from the one followed in SVR. In this case, a particular probability model for the support vectors is assumed and can be constrained to be sparse. In addition, it has been claimed that RVMs can produce probabilistic outputs (which theoretically permits to capture uncertainty in the predictions), RVMs are less sensitive to hyper-parameters setting than SVR, and the *kernel* function must not necessarily fulfil Mercer's conditions.

Once the kernel has been defined, and a particular Gaussian likelihood assumed for the target vector $\mathbf{y} = [y_1, \ldots, y_n]^\top$ given the weights $\mathbf{w}$, a maximum likelihood approach could be used for estimating model weights. However, a certain risk of overfitting arises and *a priori* models of weight distribution are commonly used in the Bayesian framework [60]. In the RVM learning scheme, rather than attempting to make sample-based (or point) predictions, a Gaussian *prior* distribution of zero mean and variance $\sigma^2_{w_j} \equiv \alpha_j^{-1}$ is defined over each weight:

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{j=1}^{n} \mathcal{N}(w_j|0, \alpha_j^{-1}) = \prod_{j=1}^{n} \sqrt{\frac{\alpha_j}{2\pi}} \exp\Big(-\frac{1}{2}\alpha_j \mathbf{w}_j^2\Big), \tag{1.40}$$

where the key to obtain sparsity is the use of n independent hyperparameters $\boldsymbol{\alpha} = (\alpha_o, \alpha_1, \ldots, \alpha_n)^\top$, one per weight (or basis function), which moderate the strength of the *prior*. After defining the *prior* over the weights, we must define the hyperpriors over $\boldsymbol{\alpha}$ and the other model parameter, the noise variance σ_n^2. These quantities were originally proposed to be Gamma distributions [59].

Now, with the *prior* (1.40) and the likelihood distribution, the posterior distribution over the weights is Gaussian and can be computed using Bayes' rule:

$$p(\mathbf{w}|\mathbf{y},\boldsymbol{\alpha},\sigma_n^2) = \frac{p(\mathbf{y}|\mathbf{w},\sigma_n^2)p(\mathbf{w}|\boldsymbol{\alpha})}{p(\mathbf{y}|\boldsymbol{\alpha},\sigma_n^2)} \sim \mathcal{N}(\mathbf{w}|\boldsymbol{\mu},\boldsymbol{\Sigma}) \tag{1.41}$$

where the covariance and the mean are respectively given by $\boldsymbol{\Sigma} = (\sigma_n^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \mathbf{A})^{-1}$ and $\boldsymbol{\mu} = \sigma_n^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top\mathbf{y}$, with $\mathbf{A} = \mathrm{diag}(\boldsymbol{\alpha})$. Hence, the Gaussian likelihood distribution over the training targets can be "marginalized" by integrating out the weights to obtain the *marginal likelihood* for the hyperparameters:

$$p(\mathbf{y}|\boldsymbol{\alpha},\sigma_n^2) = \int p(\mathbf{y}|\mathbf{w},\sigma_n^2)p(\mathbf{w}|\boldsymbol{\alpha})d\mathbf{w} \sim \mathcal{N}(0,\mathbf{C}) \tag{1.42}$$

where the covariance is given by $\mathbf{C} = \sigma_n^2\mathbf{I} + \boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^\top$. For computational efficiency, the logarithm of the evidence is maximized:

$$\mathcal{L}(\boldsymbol{\alpha}) = \log p(\mathbf{y}|\boldsymbol{\alpha},\sigma_n^2) = -\frac{1}{2}\left(n\log 2\pi + \log|\mathbf{C}| + \mathbf{y}^\top\mathbf{C}^{-1}\mathbf{y}\right), \tag{1.43}$$

which is commonly done using the standard *type-II maximum likelihood procedure*. However, [59] did not suggest direct minimization of the negative log evidence for training the RVM, but rather the use of an approximate Expectation-Maximization (EM) procedure [61].

In the RVM learning scheme, the estimated value of model weights is given by the mean of the posterior distribution (1.41), which is also the *maximum a posteriori* (MAP) estimate of the weights. The MAP estimate of the weights depends on the value of hyperparameters $\boldsymbol{\alpha}$ and the noise σ_n^2. The estimate of these two variables ($\hat{\boldsymbol{\alpha}}$ and $\hat{\sigma}_n^2$) is obtained by maximizing the marginal likelihood (1.42). The uncertainty about the optimal value of the weights reflected by the posterior distribution (1.41) is used to express uncertainty about the predictions made by the model as follows. Given a new input $\mathbf{x}_*$, the probability distribution of the corresponding output y_* is given by the (Gaussian) predictive distribution:

$$p(\mathbf{y}_*|\mathbf{x}_*,\hat{\boldsymbol{\alpha}},\hat{\sigma}_n^2) = \int p(\mathbf{y}_*|\mathbf{x}_*,\mathbf{w},\hat{\sigma}_n^2)p(\mathbf{w}|\mathbf{y},\hat{\boldsymbol{\alpha}},\hat{\sigma}_n^2)d\mathbf{w} \sim \mathcal{N}(y_*,\sigma_*^2) \tag{1.44}$$

where the mean and the variance (uncertainty) of the prediction are $y_* = (\boldsymbol{\Phi})_{i,:}\ \boldsymbol{\mu}$ and $\sigma_*^2 = \hat{\sigma}_n^2 + (\boldsymbol{\Phi})_{i,:}\ \boldsymbol{\Sigma}\ (\boldsymbol{\Phi})_{i,:}^\top$.

In the iterative maximization of $\mathcal{L}(\boldsymbol{\alpha})$, many of the hyperparameters α_j tend to infinity, yielding *a posterior* distribution (1.41) of the corresponding weight w_j that tends to be a delta function centered around zero. The corresponding weight is thus deleted from the model, as well as its associated basis function, $\boldsymbol{\phi}_j(\mathbf{x})$. In the RVM framework, each basis function $\boldsymbol{\phi}_j(\mathbf{x})$ is associated to a training sample $\mathbf{x}_j$ so that $\boldsymbol{\phi}_j(\mathbf{x}) = K(\mathbf{x}_j,\mathbf{x})$. The model is built on the few training examples whose associated hyperparameters do not go to infinity during the training process, leading to a sparse solution. These examples are called the Relevance Vectors (RVs), resembling the SVs in the SVM framework.

1.4.3 Gaussian Processes (GP)

An important concern about the suitability of RVM Bayesian algorithms in biophysical parameter estimation was raised: oversparseness was easily obtained due to the use of an improper prior, which led to inaccurate predictions and poor predictive variance estimations outside the support. Recently, the introduction of Gaussian Processes (GPs) has alleviated the aforementioned problem at the cost of providing non-sparse models [62]. GPs are also a Bayesian approach to non-parametric kernel learning. Very good numerical performance and stability has been reported in remote sensing parameter retrieval [63, 17].

Gaussian processes for regression define a distribution over functions $f : \mathcal{X} \to \mathbb{R}$ fully described by a mean $m : \mathcal{X} \to \mathbb{R}$ and a covariance (kernel) function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and $K(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))^\top (f(\mathbf{x}') - m(\mathbf{x}'))]$. Hereafter we set m to be the zero function for the sake of simplicity. Now, given a finite labeled samples dataset $\{\mathbf{x}_1, \ldots \mathbf{x}_n\}$ we first compute its covariance matrix $\mathbf{K}$ in the same way as done for the Gram matrix in SVM. The covariance matrix defines a distribution over the vector of output values $f_\mathbf{x} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^\top$, such that $f_\mathbf{x} \sim \mathcal{N}(\mathbf{0}; \mathbf{K})$, which is a multivariate Gaussian distribution. Therefore the specification of the covariance function implies the form of the distribution over the functions. The role of the covariance for GPs is the same as the role of kernels in SVM, both specify the notion of similarity in the space of functions.

For training purposes, we assume that the observed variable is formed by noisy observations of the true underlying function $y = f(\mathbf{x}) + \epsilon$. Moreover we assume the noise to be additive independently and identically Gaussian distributed with zero mean and variance σ_n^2. Let us define the stacked output values $\mathbf{y} = (y_1, \ldots, y_n)^\top$, the covariance terms of the test point $\mathbf{K}_i = (K(\mathbf{x}_i, \mathbf{x}_1), \ldots, K(\mathbf{x}_i, \mathbf{x}_n))^\top$, and $K_{ii} = K(\mathbf{x}_i, \mathbf{x}_i)$. From the previous model assumption, the output values are distributed according to:

$$\begin{pmatrix} \mathbf{y} \\ f(\mathbf{x}_i) \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K} + \sigma_n^2 \mathbf{I} & \mathbf{K}_i \\ \mathbf{K}_i^\top & K_{ii} \end{pmatrix}\right) \tag{1.45}$$

For prediction purposes, the GP is obtained by computing the conditional distribution $f(\mathbf{x}_i)|\mathbf{y}, \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}; \mathbf{x}_i$, which can be shown to be a Gaussian distribution with predictive mean $\mathbf{K}_i^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$ and predictive variance $K_{ii} - \mathbf{K}_i^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{K}_i$. Therefore, two hyperparameters must be optimized: the kernel $\mathbf{K}$ and the noise variance σ_n^2.

Note that the GP mean predictor yields exactly the same solution that the obtained in the context of kernel ridge regression (i.e. unconstrained kernel regression with squared loss function and Tikhonov's regularization). Even more important is the fact that not only a mean prediction is obtained for each sample but a full distribution over the output values including an uncertainty of the prediction.

The optimization of GP hyperparameters can be done through standard cross-validation tecniques. However, a good property of the GP framework is the possibility to optimize all involved hyperparameters, $\boldsymbol{\theta}$, iteratively through gradient-descent. This is done by maximizing the negative log marginal likelihood, $p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})$, and its partial derivatives w.r.t. the hyperparameters:[3]

$$\begin{aligned}\frac{\partial \log p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}{\partial \theta_j} &= \frac{1}{2}\mathbf{y}^\top \mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta_j}\mathbf{K}^{-1}\mathbf{y} \\ -\frac{1}{2}\mathrm{Tr}\left\{\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta_j}\right\} &= \frac{1}{2}\mathrm{Tr}\left\{(\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \mathbf{K})^{-1}\frac{\partial \mathbf{K}}{\partial \theta_j}\right\},\end{aligned} \tag{1.46}$$

where $\boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{y}$, which is only computed once. This optimization is done by a particular gradient-based optimization, resulting in a relatively fast method that scales well for less than a few thousand training samples [62]. This technique not only avoids running heuristic cross-validation methods but also optimizing very flexible kernel functions and estimating the noise variance consistently.

1.4.4 Experimental Results

In this section, we evaluate the performance of SVR, RVM and GP in the estimation of oceanic chlorophyll-a concentration from measured reflectances. We compare the models in terms of accuracy, bias, and sparsity. We use the SeaBAM dataset [64], which gathers 919 *in-situ* measurements of chlorophyll concentration around the United States and Europe. The dataset contains *in situ* pigments and remote sensing reflectance measurements at wavelengths present in the SeaWiFS sensor.[4]

Developing a SVR requires selecting the following free parameters: σ (varied between 0.1 and 30), C (varied logarithmically between 10^{-2} and 10^{5}), and ε (varied logarithmically between 10^{-6} and 10^{-1}). For the case of the RVM algorithm, the σ was logarithmically varied between 0.1 and 30. For the GP, we used a scaled anisotropic RBF kernel, $K(\mathbf{x},\mathbf{x}') = \nu \exp(-\sum_{d=1}^{D} 0.5\sigma_d^{-2}(\mathbf{x}^{(d)} - \mathbf{x}^{(d)\prime})^2) + \sigma_n^2\delta_{\mathbf{x}\mathbf{x}'}$, where ν is a kernel scaling factor accounting for signal variance, D is the data input dimension (d indicates dimension), σ_d is a dedicated lengthscale for feature d, and σ_n is the magnitude of the independent noise component. It is worth noting that in order to obtain a good set of optimal parameters, a cross-validation methodology must be followed. The available data were randomly split into two sets: 460 samples for cross-validation and the remaining 459 samples for testing performance. Before training, data were centered and transformed logarithmically, as in [65].

Table 1.2 presents results in the test set for SVR, RVM and GP models. For comparison purposes, we include results obtained with a feedforward neural network trained with back-propagation (NN-BP), which is a standard

[3] $\log p(\mathbf{y}|\mathbf{x}) \equiv \log p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^\top(\mathbf{K}+\sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log(det(\mathbf{K}+\sigma_n^2\,\mathbf{I})) - \frac{n}{2}\log(2\pi)$.

[4] More information about the data can be obtained from `http://seabass.gsfc.nasa.gov/seabam/seabam.html`.

Table 1.2. Mean error (ME), root mean-squared error (RMSE), mean absolute error (MAE), and correlation coefficient between the actual and the estimated Chl-a concentration (r) of models in the test set.

	ME	**RMSE**	**MAE**	**r**	**[%]SVs/RVs**
Morel-1 †,	-0.023	0.178	0.139	0.956	–
Ocean Chlorophyll 2, OC2	-0.031	0.169	0.133	0.960	–
NN-BP, 4 hidden nodes	-0.046	0.143	0.111	0.971	–
ε-SVR in this paper	-0.070	0.139	0.105	0.971	44.3%
RVM	**-0.009**	0.146	0.107	0.970	**4.9%**
GP	**-0.009**	0.103	0.107	0.961	–

approach in biophysical parameters retrieval. Also, we include results for the model Morel-1, and the final SeaWiFS chlorophyll-a algorithm OC2 from [66]. We can observe that (i) SVR, RVM and GP show a better performance than empirical Morel and OC2 models, and also better than artificial neural networks (NN-BP); (ii) the SVR and GP techniques are more accurate (RMSE, MAE); (iii) RVM and GP are less biased (ME) than the rest of the models, and in the case of the RVMs, drastically much more sparse (only 4.9% of training samples were necessary to attain good generalization capabilities). Comparing SVR and RVM, we can state that RVMs provide accurate estimations (similar to SVR) with small number of relevant vectors. GP provides more accurate results than SVR and RVM.

1.5 Kernel Methods for Feature extraction

The curse of dimensionality refers to the problems associated with multivariate data analysis as the dimensionality increases. This problem is specially relevant in remote sensing since, as long as new technologies improve, the number of spectral bands is continuously increasing. There are two main implications of the curse of dimensionality, which critically affect pattern recognition applications in remote sensing: there is an exponential growth in the number of examples required to maintain a given sampling density (e.g. for a density of n examples *per* bin with d dimensions, the total number of examples should be n^d); and there is an exponential growth in the complexity of the target function (e.g. a density estimate or a classifier) with increasing dimensionality. In these cases, feature extraction methods are used to create a subset of new features by combinations of the existing features. Even though the use of linear methods such as principal component analysis (PCA) or partial least squares (PLS) is quite common, recent advances to cope with nonlinearities in the data based on multivariate kernel machines have been presented [67]. In the rest of the section we will briefly review the linear and nonlinear kernel versions of PCA and PLS.

1.5.1 Mutivariate Analysis Methods (MVA)

The family of multivariate analysis (MVA) methods comprises several algorithms for feature extraction that exploit correlations between data representation in input and output spaces, so that the extracted features can be used to predict the output variables, and viceversa.

Notationally, a set of training pairs $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$, with $\mathbf{x}_i \in \mathbb{R}^N$, $\mathbf{y}_i \in \mathbb{R}^M$, where $\mathbf{x}$ are the observed explanatory variables in the input space (i.e. spectral channels or bands) and $\mathbf{y}$ are the target variables in the output space (e.g. class material or corresponding physical parameter), are given. This can be also expressed using matrix notation, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^\top$, where superscript $^\top$ denotes matrix or vector transposition. $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ denote the centered versions of $\mathbf{X}$ and $\mathbf{Y}$, respectively, while $\mathbf{C}_{xx} = \frac{1}{n}\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}$ represents the covariance matrix of the input data, and $\mathbf{C}_{xy} = \frac{1}{n}\tilde{\mathbf{X}}^\top\tilde{\mathbf{Y}}$ the covariance between the input and output data.

Feature extraction is typically used before the application of machine learning algorithms to discard irrelevant or noisy components, and to reduce the dimensionality of the data, what helps also to prevent numerical problems (e.g., when $\mathbf{C}_{xx}$ is rank deficient). Linear feature extraction can be carried out by projecting the data into the subspaces characterized by projection matrices $\mathbf{U}$ and $\mathbf{V}$, of sizes $N \times n_p$ and $M \times n_p$, so that the n_p extracted features of the original data are given by $\tilde{\mathbf{X}}' = \tilde{\mathbf{X}}\mathbf{U}$ and $\tilde{\mathbf{Y}}' = \tilde{\mathbf{Y}}\mathbf{V}$.

Principal Component Analysis (PCA)

Principal component analysis [68], also known as the *Hotelling transform* or the *Karhunen-Loeve transform*, projects linearly the input data onto the directions of largest input variance. To perform PCA, the covariance matrix is first estimated $\mathbf{C}_{xx} = 1/n \sum_{i=1}^n \tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_i^\top$. Then, the eigenvalue problem $\mathbf{C}_{xx}\mathbf{u}_i = \lambda_i\mathbf{u}_i$ is solved, which yields a set of sorted eigenvalues $\{\lambda_i\}_{i=1}^{n_p}$ $(\lambda_i \leq \lambda_{i+1})$ and the corresponding eigenvectors $\{\mathbf{u}_i\}_{i=1}^{n_p}$. Finally, new data are projected onto the eigenvectors with largest eigenvalues $\tilde{\mathbf{X}}' = \tilde{\mathbf{X}}\mathbf{U}$.

This can also be expressed more compactly as:

$$\begin{aligned} \text{PCA:} \qquad & \mathbf{U} = \arg\max_{\mathbf{U}} \ \operatorname{Tr}\{\mathbf{U}^\top\mathbf{C}_{xx}\mathbf{U}\} \\ & \text{subject to: } \ \mathbf{U}^\top\mathbf{U} = \mathbf{I} \end{aligned} \tag{1.47}$$

where $\mathbf{I}$ is the identity matrix of size $n_p \times n_p$. Using Lagrange multipliers, it can be shown (see, e.g. [19]) that the solution to (1.47) is given by the singular value decomposition (SVD) of $\mathbf{C}_{xx}$.

The main limitation of PCA is that it does not consider class separability since it does not take into account the target variables $\mathbf{y}$ of the input vectors. PCA simply performs a coordinate rotation that aligns the transformed axes with the directions of maximum variance of the original data distribution. Thus, there is no guarantee that the directions of maximum variance will contain good features for discrimination or regression.

Partial Least Squares (PLS)

Partial least squares [69] assumes that the system of interest is driven by a few latent variables (also called factors or components), which are *linear* combinations of observed explanatory variables (spectral bands). The underlying idea of PLS is to exploit not only the variance of the inputs but also their covariance with the target, which is presumably more important.

The goal of PLS is to find the directions of maximum covariance between the projected input and output data:

$$\begin{aligned} \text{PLS:} \qquad & \mathbf{U},\mathbf{V} = \underset{\mathbf{U},\mathbf{V}}{\arg\max}\ \ \mathrm{Tr}\{\mathbf{U}^\top \mathbf{C}_{xy}\mathbf{V}\} \\ & \text{subject to:}\ \ \mathbf{U}^\top\mathbf{U} = \mathbf{V}^\top\mathbf{V} = \mathbf{I} \end{aligned} \tag{1.48}$$

The solution to this problem is given by the singular value decomposition of $\mathbf{C}_{xy}$.

1.5.2 Kernel Multivariate Analysis (KMVA)

All previous methods assume that there exists a *linear* relation between the original data matrices, $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$, and the extracted projections, $\tilde{\mathbf{X}}'$ and $\tilde{\mathbf{Y}}'$, respectively. However, in many situations this linearity assumption is not satisfied, and nonlinear feature extraction is needed to obtain acceptable performance. In this context, *kernel methods* are a promising approach, as they constitute an excellent framework to formulate nonlinear versions from linear algorithms [5, 19]. In this section, we describe the kernel PCA (KPCA) and kernel PLS (KPLS) implementations.

Notationally, data matrices for performing the linear feature extraction (PCA or PLS) in $\mathcal{H}$ are now given by $\boldsymbol{\Phi} = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)]^\top$ and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^\top$. As before, the centered versions of these matrices are denoted by $\tilde{\boldsymbol{\Phi}}$ and $\tilde{\mathbf{Y}}$.

Now, the projections of the input and output data will be given by $\tilde{\boldsymbol{\Phi}}' = \tilde{\boldsymbol{\Phi}}\mathbf{U}$ and $\tilde{\mathbf{Y}}' = \tilde{\mathbf{Y}}\mathbf{V}$, respectively, where the projection matrix $\mathbf{U}$ is now of size $\dim(\mathcal{H}) \times n_p$. Note, that the input covariance matrix in $\mathcal{H}$, which is usually needed by the different MVA methods, becomes of size $\dim(\mathcal{H}) \times \dim(\mathcal{H})$ and cannot be directly computed. However, making use of the *representer's theorem* [19],we can introduce $\mathbf{U} = \tilde{\boldsymbol{\Phi}}^\top \mathbf{A}$ into the formulation, where $\mathbf{A} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{n_p}]$ and $\boldsymbol{\alpha}_i$ is an n-length column vector containing the coefficients for the ith projection vector, and the maximization problem can be reformulated in terms of the kernel matrix.

Note that, in these kernel feature extraction methods, the projection matrix $\mathbf{U}$ in $\mathcal{H}$ might not be explicitly calculated, but the projections of the input data can be obtained. Therefore, the extracted features for a new input pattern $\mathbf{x}_*$ are given by:

$$\tilde{\boldsymbol{\phi}}'(\mathbf{x}_*) = \tilde{\boldsymbol{\phi}}(\mathbf{x}_*)\mathbf{U} = \tilde{\boldsymbol{\phi}}(\mathbf{x}_*)\tilde{\boldsymbol{\Phi}}^\top \mathbf{A} = \begin{bmatrix} \tilde{K}(\mathbf{x}_1, \mathbf{x}_*) \\ \vdots \\ \tilde{K}(\mathbf{x}_n, \mathbf{x}_*) \end{bmatrix} \mathbf{A} \tag{1.49}$$

which is expressed in terms of the inner products in the centered feature space (see Section 1.2.3).

Kernel Principal Component Analysis (KPCA)

As in the linear case, the aim of KPCA is to find directions of maximum variance of the input data in $\mathcal{H}$, which can be obtained by replacing $\tilde{\mathbf{X}}$ by $\tilde{\boldsymbol{\Phi}}$ in (1.47), i.e. by replacing $\mathbf{C}_{xx}$ by $\tilde{\boldsymbol{\Phi}}^\top \tilde{\boldsymbol{\Phi}}$:

$$\begin{aligned} \text{KPCA:} \qquad & \mathbf{U} = \arg\max_{\mathbf{U}} \ \operatorname{Tr}\{\mathbf{U}^\top \tilde{\boldsymbol{\Phi}}^\top \tilde{\boldsymbol{\Phi}} \mathbf{U}\} \\ & \text{subject to: } \ \mathbf{U}^\top \mathbf{U} = \mathbf{I} \end{aligned} \tag{1.50}$$

Making use of the representer's theorem one can introduce $\mathbf{U} = \tilde{\boldsymbol{\Phi}}^\top \mathbf{A}$ into the previous formulation, and the maximization problem can be reformulated as follows:

$$\begin{aligned} \text{KPCA:} \qquad & \mathbf{A} = \arg\max_{\mathbf{A}} \ \operatorname{Tr}\{\mathbf{A}^\top \tilde{\mathbf{K}}_x \tilde{\mathbf{K}}_x \mathbf{U}\} \\ & \text{subject to: } \ \mathbf{A}^\top \tilde{\mathbf{K}}_x \mathbf{A} = \mathbf{I} \end{aligned} \tag{1.51}$$

where we have defined the symmetric centered kernel matrix $\tilde{\mathbf{K}}_x = \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{\Phi}}^\top$ containing the inner products between any two points in the feature space.

The solution to the above problem can be obtained from the singular value decomposition of $\tilde{\mathbf{K}}_x \tilde{\mathbf{K}}_x$ represented by $\tilde{\mathbf{K}}_x \tilde{\mathbf{K}}_x \boldsymbol{\alpha} = \lambda \tilde{\mathbf{K}}_x \boldsymbol{\alpha}$, which has the same solution as $\tilde{\mathbf{K}}_x \boldsymbol{\alpha} = \lambda \boldsymbol{\alpha}$.

Kernel Partial Least Squares (KPLS)

As in the linear case, the aim of KPLS is to find directions of maximum covariance between the input data in $\mathcal{H}$ and $\mathbf{Y}$, and can thus be expressed as:

$$\begin{aligned} \text{KPLS:} \qquad & \mathbf{U}, \mathbf{V} = \arg\max_{\mathbf{U},\mathbf{V}} \ \operatorname{Tr}\{\mathbf{U}^\top \tilde{\boldsymbol{\Phi}}^\top \tilde{\mathbf{Y}} \mathbf{V}\} \\ & \text{subject to: } \ \mathbf{U}^\top \mathbf{U} = \mathbf{V}^\top \mathbf{V} = \mathbf{I} \end{aligned} \tag{1.52}$$

Again, making use of the representer's theorem, one can introduce $\mathbf{U} = \tilde{\boldsymbol{\Phi}}^\top \mathbf{A}$ into the previous formulation, and the maximization problem can be reformulated as follows:

$$\begin{aligned} \text{KPLS:} \qquad & \mathbf{A}, \mathbf{V} = \arg\max_{\mathbf{A},\mathbf{V}} \ \operatorname{Tr}\{\mathbf{A}^\top \tilde{\mathbf{K}}_x \tilde{\mathbf{Y}} \mathbf{V}\} \\ & \text{subject to: } \ \mathbf{A}^\top \tilde{\mathbf{K}}_x \mathbf{A} = \mathbf{V}^\top \mathbf{V} = \mathbf{I} \end{aligned} \tag{1.53}$$

The solution to the above problem can be obtained from the singular value decomposition of $\tilde{\mathbf{K}}_x \tilde{\mathbf{Y}}$.

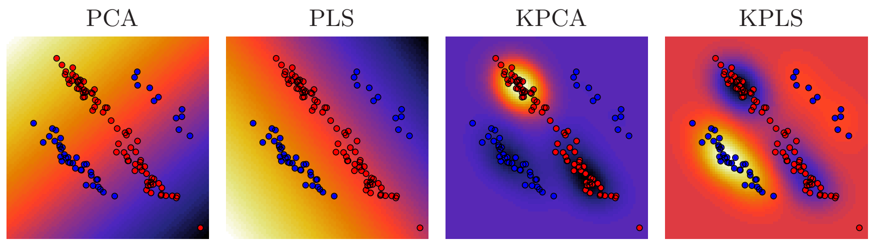

Fig. 1.4. First extracted component by linear (PCA, PLS) and nonlinear kernel (KPCA, KPLS) methods.

1.5.3 Experimental Results

Figure 1.4 illustrates the performance of linear and kernel MVA feature extraction methods in a 2D toy example. KPCA and KPLS used an RBF kernel with the same sigma value fixed to the mean distance among all training samples. It can be observed that linear methods (PCA and PLS) cannot cope with the non-linearly separable problem, while kernel methods accommodate data relations in the kernel and define local boundaries. Performance of KPLS results in more accurate boundaries and perfectly separates the two classes, while KPCA fails as no class label information is used. Results in remote sensing image classification are reported in [70, 67].

1.6 Future Trends in Remote Sensing Kernel Learning

Even though the chapter presented an updated literature review, new kernel-based learning methodologies are being constantly explored. The special peculiarities of the acquired images lead to develop new methods. And viceversa, the new learning paradigms available offer new ways of looking at old, yet unsolved, problems in remote sensing. In what follows, we review recent research directions in the context of remote sensing kernel-based learning.

1.6.1 Multiple Kernel Learning

Composite kernels have been specifically designed and applied for the efficient combination of multitemporal, multisensor and multisource information [9, 71]. The previous approaches exploited some properties of kernel methods (such as the direct sum of Hilbert spaces, see Section 1.2.3) to combine kernels dedicated to process different signal sources, e.g. the sum of a kernel on spectral feature vectors can be summed up to a kernel defined over spatially-extracted feature vectors. This approach yielded very good results but it was limited to the combination of few kernels [26], as the optimization of kernel parameters was an issue. Lately, the composite framework approach has been extended to the framework of multiple kernel learning (MKL) [72]. In MKL, the SVM kernel function is defined as a weighted linear combination of kernels built using subsets of features. MKL works iteratively optimizing both the individual weights and the kernel parameters [73]. So far, the only application in remote sensing of strict MKL can be found in [74] and, taking advantage of a similar idea, spectrally weighted kernels are proposed in [75]. Not only a certain gain in accuracy is observed but also the final model yields some insight in the problem. In [46], the relevant features of remote sensing images for automatic classification are studied through this framework.

1.6.2 Domain Learning

A common problem in remote sensing is that of updating land-cover maps by classifying temporal series of images when only training samples collected at one time instant are available. This is known as transfer learning or domain adaptation. This setting implies that unlabeled test examples and training examples are drawn from different domains or distributions. The problem was initially tackled with partially unsupervised classifiers, both under parametric formalisms [76] and neural networks [77]. The approach was then successfully extended to domain adaptation SVM (DASVM) [78].

A related problem is also that of classifying an image using labeled pixels from other scenes, which induces the sample selection bias problem, also known as covariance shift. Here, unlabeled test data are drawn from the same training domain, but the estimated distribution does not correctly model the true underlying distribution since the number (or the quality) of available training samples is not sufficient. These problems have been recently presented by defining mean map kernel machines that account for the dispersion of data in feature spaces [45].

1.6.3 Structured Learning

Most of the techniques revised so far assume a simple set of outputs. However, more complex output spaces can be imagined, e.g. predicting multiple labels (land use and land cover simultaneously), multi-temporal image sequences, or abundance fractions. Such complex output spaces are the topic of structured learning, one of the most recent developments in machine learning. Only a computer vision application [79] and the preliminary results in [80] have been presented for image processing. Certainly this field of learning joint input-output mappings will receive attention in the future.

1.6.4 Active Learning

When designing a supervised classifier, the performance of the model strongly depends on the quality of the labeled information available. This constraint makes the generation of an appropriate training set a difficult and expensive task requiring extensive manual human-image interaction. Therefore, in order to make the models as efficient as possible, the training set should be kept as small as possible and focused on the pixels that really help to improve the performance of the model. Active learning aims at responding to this need, by constructing effective training sets.

In remote sensing, application of active learning methods that select the most relevant samples for training is quite recent. A SVM method for object-oriented classification was proposed in [81], while maximum likelihood classifiers for pixel-based classification was presented in [82]. Recently, this approach was extended in [83] by proposing boosting to iteratively weight the selected pixels. In [84, 85] information-based active learning was proposed for target detection, and in [86], a model-independent active learning method was proposed for very-high resolution satellite images.

1.6.5 Parallel Implementations

Kernel methods in general, and the SVM in particular, have the problem of scaling at least quadratically with the number of training samples. With the recent explosion in the amount and complexity of hyperspectral data, and with the increasing availability of very high resolution images, the number of labeled samples to train kernel classifiers is becoming a critical problem. In this scenario, parallel processing constitutes a requirement in many remote sensing missions, especially with the advent of low-cost systems such as commodity clusters and distributed networks of computers. Several efforts are being pursued to develop parallel implementations of SVMs for remote sensing data classification: boss-worker approaches [87, 88, 89] and parallelization through decomposition of the kernel matrix have been successfully explored [90].

1.7 Conclusions

Kernel methods allow us to transform almost any linear method into a non-linear one, while still operating with linear algebra. The methods essentially rely on embedding the examples into a high dimensional space where a linear method is designed and applied. Access to the mapped samples is done implicitly through kernel functions. This chapter reviewed the field of kernel machines in remote sensing data processing. The important topics of classification, model inversion, and feature extraction with kernels have been revised. The impact and development of kernel methods in this area during the last decade has been large and fruitful, overcoming some of the problems posed both by the recent satellite sensors acquired data, and the limitations of other machine learning methods. New developments are expected in the near future to encompass both remote sensing data complexity and new problem settings.

Acknowledgments

This work was partially supported by projects CICYT-FEDER TEC2009-13696, AYA2008-05965-C04-03, and CSD2007-00018. Valero Laparra acknowledges the support of a Ph.D grant from the Spanish Government BES-2007-16125.

The authors would like to thank a number of colleagues and collaborators in the field of kernel methods, and whose insight and points of view are at some extent reflected in this chapter: Dr. Devis Tuia from the University of Lausanne (Switzerland), Dr. Frédéric Ratle from Nuance Communications (Belgium), Dr. Jerónimo Arenas from the University Carlos III de Madrid (Spain), and Prof. Lorenzo Bruzzone from the University of Trento (Italy).

References

1. Richards, J.A., Jia, X.: Remote Sensing Digital Image Analysis. An Introduction. 3rd edn. Springer-Verlag, Berlin, Heidenberg (1999)
2. Hughes, G.F.: On the mean accuracy of statistical pattern recognizers. IEEE Trans. Information Theory **14** (1968) 55–63
3. Fukunaga, K., Hayes, R.R.: Effects of sample size in classifier design. IEEE Trans. Pattern Analysis and Machine Intelligence **11** (1989) 873–885
4. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, New York (1998)
5. Schölkopf, B., Smola, A.: Learning with Kernels – Support Vector Machines, Regularization, Optimization and Beyond. MIT Press Series (2002)
6. Camps-Valls, G., Bruzzone, L.: Kernel-based methods for hyperspectral image classification. IEEE Trans. Geosc. Remote Sensing **43** (2005) 1351–1362
7. Mercier, G., Girard-Ardhuin, F.: Partially supervised oil-slick detection by SAR imagery using kernel expansion. IEEE Trans. Geosc. Remote Sensing **44** (2006) 2839–2846
8. Muñoz Marí, J., Bruzzone, L., Camps-Valls, G.: A support vector domain description approach to supervised classification of remote sensing images. IEEE Trans. Geosc. Remote Sensing **45** (2007) 2683–2692
9. Camps-Valls, G., Gómez-Chova, L., Muñoz Marí, J., Martínez-Ramón, M., Rojo-Álvarez, J.L.: Kernel-based framework for multi-temporal and multi-source remote sensing data classification and change detection. IEEE Trans. Geosc. Remote Sensing **46** (2008) 1822–1835
10. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. 1st edn. MIT Press, Cambridge, Massachusetts and London, England (2006)
11. Camps-Valls, G., Bruzzone, L., Rojo-Álvarez, J.L., Melgani, F.: Robust support vector regression for biophysical variable estimation from remotely sensed images. IEEE Geoscience and Remote Sensing Letters **3** (2006) 339–343
12. Zortea, M., De Martino, M., Moser, G., Serpico, S.B.: Land surface temperature estimation from infrared satellite data using support vector machines. In: Proceedings of the IGARSS-2006 Symposium, Denver, USA (2003) 2109–2112
13. Durbh, S.S., King, R.L., Younan, N.H.: Support vector machines regression for retrieval of leaf area index from multiangle imaging spectroradiometer. Remote Sensing of Environment **107** (2007) 348–361

14. Yang, F., White, M., Michaelis, A., Ichii, K., Hashimoto, H. Votava, P., Zhu, A.X., Nemani, R.: Prediction of continental-scale evapotranspiration by combining MODIS and AmeriFlux data through support vector machine. IEEE Trans. Geosc. Remote Sensing **44** (2006) 3452–3461
15. Broadwater, J., Chellappa, R., Banerjee, A., Burlina, P.: Kernel fully constrained least squares abundance estimates. In: Proceedings of the IGARSS-2007 Symposium, Barcelona, Spain (2007)
16. Camps-Valls, G., Gomez-Chova, L., Vila-Francés, J., Amorós-López, J., Muñoz-Marí, J., Calpe-Maravilla, J.: Retrieval of oceanic chlorophyll concentration with relevance vector machines. Remote Sensing of Environment **105** (2006) 23–33
17. Pasolli, L., Melgani, F., Blanzieri, E.: Estimating biophysical parameters from remotely sensed imagery with Gaussian Processes. In: IEEE International Geoscience and Remote Sensing Symposium, IGARSS'08, Boston, USA (2008)
18. Camps-Valls, G., Gómez-Chova, L., Muñoz-Marí, J., Calpe-Maravilla, J.: Biophysical parameter estimation with adaptive Gaussian processes. In: IEEE International Geosc. Remote Sensing Symp., IGARSS'2009, Capetown, South Africa (2009)
19. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
20. Golub, G.H., Van Loan, C.F.: Matrix Computations (Johns Hopkins Studies in Mathematical Sciences). The Johns Hopkins University Press (1996)
21. Reed, M.C., Simon, B.: Functional Analysis. Volume I of Methods of Modern Mathematical Physics. Academic Press (1980)
22. Huang, C., Davis, L., Townshend, J.: An assessment of support vector machines for land cover classification. Int. J. Remote Sens. **23:4** (2002) 725–749
23. Foody, G.M., Mathur, A.: Toward intelligent training of supervised image classifications: directing training data acquisition for SVM classification. Remote Sens. Environ. **93** (2004) 107–117
24. Fauvel, M., Chanussot, J., Benediktsson, J.A.: Evaluation of kernels for multiclass classification of hyperspectral remote sensing data. In: IEEE ICASSP - International conference on Acoustics, Speech and Signal Processing, Toulouse, France (2006) II–813 – II–816
25. Inglada, J.: Automatic recognition of man-made objects in high resolution optical remote sensing images by SVM classification of geometric image features. ISPRS Journal of Photogrammetry Rem. Sens. **62** (2007) 236–248
26. Camps-Valls, G., Gómez-Chova, L., Muñoz-Marí, J., Vila-Francés, J., Calpe-Maravilla, J.: Composite kernels for hyperspectral image classification. IEEE Geoscience and Remote Sensing Letters **3** (2006) 93–97
27. Chi, M., Feng, R., Bruzzone, L.: Classification of hyperspectral remote-sensing data with primal SVM for small-sized training dataset problem. Adv. Space Res. **41 (11)** (2008) 1793–1799
28. Fauvel, M., Benediktsson, J.A., Chanussot, J., Sveinsson, J.R.: Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles. IEEE Trans. Geosci. Remote Sens. **46 (11)** (2008) 3804 – 3814
29. Schölkopf, B., Smola, A.J., Williamson, R.C., Bartlett, P.L.: New support vector algorithms (2000)
30. Tax, D., Duin, R.P.: Support vector domain description. Pattern Recognition Letters **20** (1999) 1191–1199
31. Schölkopf, B., Williamson, R.C., Smola, A., Shawe-Taylor, J.: Support vector method for novelty detection. In: Advances in Neural Information Processing Systems 12, Denver, CO (1999)

32. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annals Eugenics **7** (1936) 179–188
33. Hastie, T., Tibishirani, R., Friedman, J.: The Elements of Statistical Learning. Springer-Verlag, New York, NY (2001)
34. Gómez-Chova, L., Fernández-Prieto, D., Calpe, J., Soria, E., Vila-Francés, J., Camps-Valls, G.: Urban monitoring using multitemporal SAR and multispectral data. Pattern Recognition Letters **27** (2006) 234–243 3rd Pattern Recognition in Remote Sensing Workshop, Kingston Upon Thames, England, Aug 27, 2004.
35. Congalton, Russell G., Green, Kass: Assessing the Accuracy of Remotely Sensed data: Principles and Practices. Lewis Publishers, Boca Raton, Florida (1999)
36. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, USA (2005) Online document: http://www.cs.wisc.edu/∼jerryzhu/pub/ssl_survey.pdf.
37. Chapelle, O., Weston, J., Schölkopf, B.: Cluster kernels for semi-supervised learning. In Becker, S., Thrun, S., Obermayer, K., eds.: NIPS 2002. Volume 15., Cambridge, MA, USA, MIT Press (2003) 585–592
38. Seeger, M.: Learning with labeled and unlabeled data. Technical Report TR.2001, Institute for Adaptive and Neural Computation, University of Edinburg (2001)
39. Jackson, Q., Landgrebe, D.: An adaptive classifier design for high-dimensional data analysis with a limited training data set. IEEE Trans. Geosc. Remote Sensing (2001) 2664–2679
40. Bruzzone, L., Chi, M., Marconcini, M.: A novel transductive SVM for the semisupervised classification of remote-sensing images. IEEE Trans. Geosc. Remote Sensing **44** (2006) 3363–3373
41. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing Systems, NIPS2004. Volume 16., Vancouver, Canada, MIT Press (2004)
42. Camps-Valls, G., Bandos, T., Zhou, D.: Semi-supervised graph-based hyperspectral image classification. IEEE Trans. Geosc. Remote Sensing **45** (2007) 2044–3054
43. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. Journal of Machine Learning Research **7** (2006) 2399–2434
44. Gómez-Chova, L., Camps-Valls, G., Muñoz-Marí, J., Calpe-Maravilla, J.: Semi-supervised image classification with Laplacian support vector machines. IEEE Geoscience and Remote Sensing Letters **5** (2008) 336–340
45. Gómez-Chova, L., Camps-Valls, G., Bruzzone, L., Calpe-Maravilla, J.: Mean map kernel methods for semisupervised cloud classification. IEEE Trans. on Geoscience and Remote Sensing **48** (2010) 207–220
46. Tuia, D., Camps-Valls, G.: Semisupervised remote sensing image classification with cluster kernels. Geoscience and Remote Sensing Letters, IEEE **6** (2009) 224–228
47. Tikhonov, A.N.: Regularization of incorrectly posed problems. Sov. Math. Dokl. **4** (1963) 1624–1627
48. Evgeniou, T., Pontil, M., Poggio, T.: Regularization networks and support vector machines. Advances in Computational Mathematics **13** (2000) 1–50
49. Lillesand, T.M., Kiefer, R.W., Chipman, J.: Rem. Sens. and Image Interpretation. J. Wiley & Sons, NY (2008)
50. Kimes, D., Knyazikhin, Y., Privette, J., Abuelgasim, A., Gao, F.: Inversion methods for Physically-Based Models. Rem. Sens. Reviews **18** (2000) 381–439

51. Keiner, L.E.: Estimating oceanic chlorophyll concentrations with neural networks. International Journal of Remote Sensing **20** (1999) 189–194
52. Dzwonkowski, B., Yan, X.H.: Development and application of a neural network based colour algorithm in coastal waters. Intl. J. Rem. Sens. **26** (2005) 1175–1200
53. Camps-Valls, G., Muñoz-Marí, J., Gómez-Chova, L., Richter, K., Calpe-Maravilla, J.: Biophysical Parameter Estimation with a Semisupervised Support Vector Machine. IEEE Geoscience and Remote Sensing Letters **6** (2009) 248–252
54. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Statistics and Computing **14** (2004) 199–222
55. Kwiatkowska, E., Fargion, G.: Application of machine-learning techniques toward the creation of a consistent and calibrated global chlorophyll concentration baseline dataset using remotely sensed ocean color data. IEEE Trans. Geosc. Remote Sensing **41** (2003) 2844 – 2860
56. Zhan, H., Shi, P., Chen, C.: Retrieval of oceanic chlorophyll concentration using support vector machines. IEEE Trans. Geosc. Remote Sensing **41** (2003) 2947–2951
57. Fletcher, R.: Practical Methods of Optimization. John Wiley & Sons, Inc. 2nd Edition (1987)
58. Courant, R., Hilbert, D.: Methods of Mathematical Physics. Interscience Publications. John Wiley and Sons, New York, U.S.A. (1953)
59. Tipping, M.E.: Sparse Bayesian Learning and the Relevance Vector Machine. J. of Mach. Learn. Res. **1** (2001) 211–244
60. O'Hagan, A.: Bayesian Inference, volume 2B of Kendall's Advanced Theory of Statistics. Arnold, London, United Kingdom (1994)
61. Dempster, A. P. Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B **39** (1977) 1–38
62. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press, New York (2006)
63. Furfaro, R., Morris, R.D., Kottas, A., Taddy, M., Ganapol, B.D.: A Gaussian Process Approach to Quantifying the Uncertainty of Vegetation Parameters from Remote Sensing Observations. AGU Fall Meeting Abstracts (2006) A261
64. O'Reilly, J.E., Maritorena, S., Mitchell, B.G., Siegel, D.A., Carder, K., Garver, S.A., Kahru, M., McClain, C.: Ocean color chlorophyll algorithms for SeaWiFS. Journal of Geophysical Research **103** (1998) 24937–24953
65. Cipollini, P., Corsini, G., Diani, M., Grass, R.: Retrieval of sea water optically active parameters from hyperspectral data by means of generalized radial basis function neural networks. IEEE Trans. Geosc. Remote Sensing **39** (2001) 1508–1524
66. Maritorena, S., O'Reilly, J. In: OC2v2: Update on the initial operational SeaWiFS chlorophyll algorithm in "SeaWiFS Postlaunch Calibration and Validation Analyses". Volume 11. John Wiley & Sons, NASA Goddard Space Flight Center, Greenbelt, Maryland, USA (2000) 3–8
67. Camps-Valls, G., Bruzzone, L.: Kernel Methods for Remote Sensing Data Analysis. J. Wiley & Sons (2009)
68. Jolliffe, I.T.: Principal Component Analysis. Springer-Verlag (1986)
69. Wold, S., Albano, C., Dunn, W.J., Edlund, U., Esbensen, K., Geladi, P., Hellberg, S., Johansson, E., Lindberg, W., Sjostrom, M.: Multivariate Data Analysis in Chemistry. In: Chemometrics, Mathematics and Statistics in Chemistry. Reidel Publishing Company (1984) 17

70. Arenas-García, J., Camps-Valls, G.: Efficient kernel orthonormalized PLS for remote sensing applications. IEEE Trans. Geosc. Remote Sensing **46** (2008) 2872 – 2881
71. Tuia, D., Ratle, F., Pozdnoukhov, A., Camps-Valls, G.: Multisource composite kernels for urban image classification. IEEE Geoscience and Remote Sensing Letters (2009)
72. Lancricket, G., Bie, T.D., Cristianini, N., Jordan, M., Noble, W.: A statistical framework for genomic data fusion. Bioinformatics **20** (2004) 2626–2635
73. Sonnenburg, S., G. Rätsch, C.S., Schölkopf, B.: Large scale multiple kernel learning. Journal of Machine Learning Research **7** (2006) 1531–1565
74. Villa, A., Fauvel, M., Chanussot, J., Gamba, P., Benediktsson, J.A.: Gradient optimization for multiple kernel parameters in support vector machines classification. In: IEEE International Geoscience and Remote Sensing Symposium, IGARSS. (2008)
75. Guo, B., Gunn, S., Damper, R.I., Nelson, J.D.B.: Customizing kernel functions for SVM-based hyperspectral image classification. IEEE Trans. Im. Proc. **17** (2008) 622–629
76. Bruzzone, L., Prieto, D.: Unsupervised retraining of a maximum likelihood classifier for the analysis of multitemporal remote sensing images. IEEE Trans. Geosc. Remote Sensing **39** (2001) 456–460
77. Bruzzone, L., Cossu, R.: A multiple-cascade-classifier system for a robust and partially unsupervised updating of land-cover maps. IEEE Trans. Geosc. Remote Sensing **40** (2002) 1984–1996
78. Bruzzone, L., Marconcini, M.: Toward the automatic updating of land-cover maps by a domain-adaptation SVM classifier and a circular validation strategy. IEEE Trans. Geosc. Remote Sensing **47** (2009) 1108–1122
79. Blaschko, M., Lampert, C.: Learning to localize objects with structured output regression. In Forsyth, D., Torr, P., Zisserman, A., eds.: Computer Vision: ECCV 2008., Springer (2008) 2–15
80. Tuia, D., Kanevski, M., Muñoz Marí, J., Camps-Valls, G.: Structured SVM for remote sensing image classification. In: IEEE Workshop on Machine Learning for Signal Processing (MLSP09), Grenoble, France (2009)
81. Mitra, P., Uma Shankar, B., Pal, S.: Segmentation of multispectral remote sensing images using active support vector machines. Pattern Recogn. Lett. **25** (2004) 1067–1074
82. Rajan, S., Ghosh, J., Crawford, M.M.: An active learning approach to hyperspectral data classification. IEEE Trans. Geosci. Remote Sens. **46 (4)** (2008) 1231–1242
83. Jun, G., Ghosh, J.: An efficient active learning algorithm with knowledge transfer for hyperspectral remote sensing data. In: Proceedings of the IEEE Geosc. Remote Sensing Symp. IGARSS. (2008)
84. Zhang, C., Franklin, S., Wulder, M.: Geostatistical and texture analysis of airborne-acquired images used in forest classification. Int. J. Remote Sens. **25** (2004) 859–865
85. Liu, Q., Liao, X., Carin, L.: Detection of unexploded ordnance via efficient semisupervised and active learning. IEEE Trans. Geosci. Remote Sens. **46 (9)** (2008) 2558–2567
86. Tuia, D., Ratle, F., Pacifici, F., Kanevski, M., Emery, W.: Active learning methods for remote sensing image classification. IEEE Trans. Geosc. Remote Sensing **47** (2009) 2218–2232

87. Brazile, J., Schaepman, M.E., Schläpfer, D., Kaiser, J.W., Nieke, J., Itten, K.I.: Cluster versus grid for large-volume hyperspectral image preprocessing. In Huang, H.L.A., Bloom, H.J., eds.: Atmospheric and Environmental Remote Sensing Data Processing and Utilization: an End-to-End System Perspective. Edited by Huang, Hung-Lung A.; Bloom, Hal J. Proceedings of the SPIE, Volume 5548, pp. 48-58 (2004). Volume 5548 of Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference. (2004) 48–58
88. Gualtieri, J.A.: A parallel processing algorithm for remote sensing classification. Technical report, Summaries of the Airborne Earth Science Workshop, Pasadena, USA (2004) Available: http://aviris.jpl.nasa.gov/html/aviris/documents.html.
89. Plaza, A., Chang, C.I.: High Performance Computing in Remote Sensing. Chapman & Hall/CRC Press, Boca Raton, FL (2007)
90. Muñoz, J., Plaza, A., Gualtieri, J.A., Camps-Valls, G.: Parallel Implementation of SVM in Earth Observation Applications. In Xhafa, F., ed.: Parallel Programming and Applications in Grid, P2P and Networking systems. IOS Press, UK (2009) 292–312