Filter bank learning for signal classification

Maxime Sangnier, Jérôme Gauthier, A. Rakotomamonjy

Filter Bank Learning for Signal Classification

M. Sangnier[a,*], J. Gauthier[a], A. Rakotomamonjy[b]

[a] *CEA, LIST, 91191 Gif-sur-Yvette CEDEX, France*
[b] *Université de Rouen, LITIS EA 4108, 76800 Saint-Etienne-du-Rouvray, France*

Abstract

This paper addresses the problem of feature extraction for signal classification. It proposes to build features by designing a data-driven filter bank and by pooling the time-frequency representation to provide time-invariant features. For this purpose, our work tackles the problem of jointly learning the filters of a filter bank with a support vector machine. It is shown that, in a restrictive case (but consistent to prevent overfitting), the problem boils down to a multiple kernel learning instance with infinitely many kernels. To solve such a problem, we build upon existing methods and propose an active constraint algorithm able to handle a non-convex combination of an infinite number of kernels. Numerical experiments on both a brain-computer interface dataset and a scene classification problem prove empirically the appeal of our method.

Keywords: Signal classification, Filter Bank, SVM, Kernel learning.

1. Introduction

The problem of signal classification is now becoming more and more ubiquitous, ranging from phoneme or environmental signal to biomedical signal classification. As classifiers are often built on a geometric interpretation (the way some points are arranged in a space), the usual trend to classify signals is first to extract features from the signals, and then to feed the classifier with them. The classifier can thus learn an optimal decision function based on features extracted from some training examples. Such features are diverse according to the given classification problem: physical perceptions (loudness), statistical moments (covariance matrix), spectral characterization (Fourier transform), Time-Frequency (TF) representations (spectrograms, wavelet decompositions) and so on. In this scope, features are chosen so as to characterize similarities within a class and disparities between classes to be distinguished. This property of the features, called discrimination, obviously affects the classifier accuracy: the more discriminative the features are, the better the classifier performs. Yet, discrimination power may be a subjective concept, relative to the classifier. Contrarily to this remark, the features extractor is usually arbitrarily chosen and considered independently from the choice of the pattern recognition algorithm, so much that there is no guarantee of any classification efficiency.

Finding discriminative features with respect to the chosen classifier became a field of interest in the 1990's. It appeared in different areas for different categories of features (*e.g.* [1, 2, 3]). Particularly in signal classification, data-driven feature extraction algorithms have been especially designed for TF analysis. This choice is explained by the study of real-world signals, which are by nature transient (in most cases). For this kind of signals, keeping information in both time and frequency is of major interest. The bulk of the scientific contributions, concerning automated learning of a discriminative TF transform, can roughly be clustered in four fields: wavelet [4, 5, 6] and Cohen distribution [7, 8] design, dictionary [9, 10, 11] and Filter Banks (FBs) [12, 13] learning. The previous works exhibit different aspects of designing a data-driven TF transform: atomic (wavelets, dictionaries) *vs* bilinear decompositions (Cohen transform), convolution (filters) *vs* matrix product (dictionaries), splitting decompositions to feed different modules *vs* keeping the resulting TF representation all at once, genetic *vs* gradient-based optimization, misclassification rate *vs* risk minimization and so on. Yet, most of these references share one feature: the classifier used is based on Support Vector Machines (SVMs).

As for us, we are interested in the *convolution – all at once* approach, meaning that we will consider filters rather than dictionaries and a single SVM instead of a complex classifier (our motivation being the sake of simplicity and computational efficiency). Concerning signal processing, this brings us to focus on FBs. This choice is motivated by the ability of FBs to model a wide class of atomic decompositions (for instance the cosine, the short-time Fourier and the wavelet transforms). This ability to analyze signals in the TF domain is particularly suited for transient signals and has thus motivated a couple of decades of intensive studies about FBs from the viewpoint of the reconstructive approach (with applications in denoising and compression)

*Corresponding author.
Email addresses: maxime.sangnier@gmail.com (M. Sangnier), jerome.gauthier@cea.fr (J. Gauthier), alain.rakotomamonjy@univ-rouen.fr (A. Rakotomamonjy)

[14, 15, 16]. As far as we are concerned, FBs reappeared recently to be adequate for discriminative feature extraction [17, 18, 19]. Another motivation for choosing FBs is that they enable to keep a strong link with the signal processing approach, as much as with the machine learning part, and thus to provide experts with interpretable signal processing tools. Indeed the very definition of FBs, which are arrays of filters [14], ensures a direct TF interpretation.

One more point about the classification process has to be noticed: the TF representation of a signal is rarely considered in itself as a classification feature, for TF representations highly depend on random time shifts. This property is damaging for the recognition of patterns within the signals. This is one of the drawbacks resulting from keeping information both in frequency and time. As a consequence, final classification features are usually obtained by performing a sort of aggregation of the data obtained from the TF transform. This operation is called *pooling* [12] and will be detailed further in the manuscript. In the end, the processing chain is successively made up of a TF representation, a pooling function and a classifier. Our objective, in this work is thus to propose a novel methodology to jointly learn the features obtained from an FB together with an SVM classifier, thanks to solving an optimization problem.

In a learning task posed as an optimization problem, the cost function and the solving method are part of the central concerns. Very often and aside from the dictionary learning, the answers boil down to a gradient-based optimization or to an evolutionary algorithm minimizing the misclassification rate on a random evaluation dataset [12, 5, 13]. In this work, we prefer to adopt a Structural Risk Minimization approach [20]. This is the one used in Support Vector Machines. The main benefits of this approach are to be based on convex optimization (rather than non-convex and thus difficult) and to consider a regularized measure of the misclassification rate, that tends to prevent overfitting.

This work extends the results given in a previous conference paper [21]. Our contributions to the state of the art stand on the following points:

- we introduce a novel framework which casts the problem of jointly learning features extracted from an FB and a large-margin classifier (SVM);
- we show that by restricting in some sense the set of possible FBs, the optimization problem we have to solve boils down to a generalized version of a Multiple Kernel Learning problem (MKL);
- because the optimization problem we are interested in involves an infinite number of possible filters, we extend existing non-linear MKL algorithms to let them handle such a situation;
- interestingly, our framework allows us to also learn the pooling function which builds the final features from the TF representation obtained by the FB. As far as we know, this is one of the first successful attempt at learning the pooling function.

The paper is organized as follows: considering some notations (Table 1, page 2), we first describe the problem of FB learning, after reminding the basis of FBs and giving details of some interesting pooling functions. Then, we expose the restricted framework in which our work holds and describe the proposed algorithm, which handles an infinite amount of filters. We also give some insights concerning a way to learn the pooling function. As a third part, this paper discusses some details of the proposed algorithm. Then, numerical results are exposed in order to back up the appeal of this approach. Binary classification problems based on three different datasets are tackled: a built toy dataset, Brain-Computer Interface (BCI) signals and environmental audio scenes. Eventually, the last section deals with the comparison of our approach with previous works, including Infinite Kernel Learning (IKL), Wavelet Kernel Learning (WKL) and Convolutional Neural Networks (CNNs).

Symbol	*Definition*
$\imath$	Imaginary unit.
$\mathbb{1}$	Indicator vector (its size depends on the context).
$\preccurlyeq, \succcurlyeq$	Pointwise inequalities.
$\lvert\cdot\rvert$	Pointwise modulus.
$\circ$	Depending on the context, the Hadamard product between two matrices or composition between two functions.
$\star$	Convolution.
$\mathcal{X}$	Set of signals: $\mathcal{X} = \cup_{n=1}^{\infty} \mathbb{K}^n$.
$\mathbb{K}$	Divising ring (either $\mathbb{R}$ or $\mathbb{C}$).
$\boldsymbol{K}_+$	Positive semidefinite matrix.

Table 1: Definitions.

2. Filtering to improve signal recognition

This section introduces the learning framework we are interested in. After reviewing briefly FBs, we highlight the several manners, we use in this study, to pool a TF representation and then describe the learning problem we want to address.

2.1. Filter bank: definitions and notations

FBs are models of linear transformations that embody a large class of signal processing transforms, among which the discrete Fourier, cosine and wavelet transforms. Conceptually, an FB is an array of filters followed by downsampling (or decimation) operations (Figure 1) [22]. A quite famous example is the Fast Wavelet Transform [23], which is implemented as a multistage FB (*i.e.* as a cascade of FBs). It can yet be redrawn as a one-stage bank of $l + 1$

filters (with l the scale of the discrete wavelet transform), as shown by the example in Figure 2.

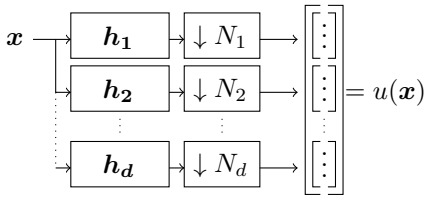
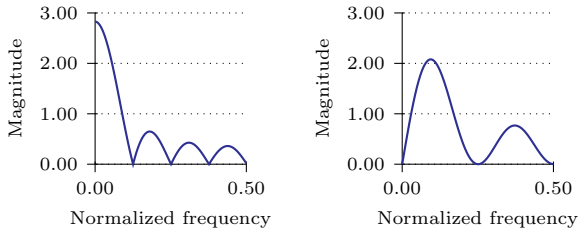
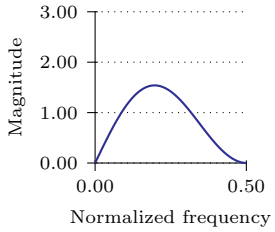


Figure 1: Diagram of a d-channel filter bank u.

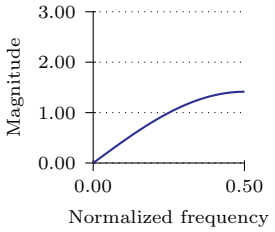


(a) Filter $\boldsymbol{h_1}$ ($N_1 = 8$).

(b) Filter $\boldsymbol{h_2}$ ($N_2 = 8$).



(c) Filter $\boldsymbol{h_3}$ ($N_3 = 4$).



(d) Filter $\boldsymbol{h_4}$ ($N_4 = 2$).

Figure 2: Filter bank of a Haar wavelet transform at scale 3 (in the Fourier domain).

Mathematically, an FB $u\colon \mathcal{X} \to \mathcal{X}^d$ is a linear application defined by a number d of linear filters and of decimation factors $\{N_l\}_{1\leq l\leq d}$. In this study, the filters are parametrized by their finite Impulse Responses (IRs) $\{\boldsymbol{h_l}\}_{1\leq l\leq d}$ of respective lengths $\{q_l\}_{1\leq l\leq d}$. Consequently, we also note $u = (\boldsymbol{h_l}, N_l)_{1\leq l\leq d}$ to express this parametrization. A formal definition of u is:

$$\begin{aligned}&\forall \boldsymbol{x} \in \mathbb{K}^n\colon\\ &u(\boldsymbol{x}) \stackrel{\text{def}}{=} \{\downarrow N_l\,[\boldsymbol{h_l} \star \boldsymbol{x}]\}_{1\leq l\leq d}\\ &\qquad\stackrel{\text{def}}{=} \left\{\downarrow N_l \left[\left(\sum_{j=1}^{q_l} h_l(j)\tilde{x}(i-j+1)\right)_{1\leq i\leq n}\right]\right\}_{1\leq l\leq d},\end{aligned} \tag{1}$$

where $\tilde{\boldsymbol{x}}$ is the signal $\boldsymbol{x}$ extended to the past (through periodization, symmetrization or zero-padding) and $\downarrow N_l$ is the downsampling operator:

$$\forall \boldsymbol{x} \in \mathbb{K}^n\colon\ \downarrow N[\boldsymbol{x}] \stackrel{\text{def}}{=} (x(1+N(j-1)))_{1\leq j\leq \lfloor \frac{n}{N} \rfloor}.$$

Note that with the definition (1), $u(\boldsymbol{x})$ is a set of filtered and downsampled signals, which embodies a TF representation.

FBs have been extensively studied through the polyphase framework for the purpose of denoising and compression [14]. More precisely, given an invertible FB similar to the one in Figure 1 (the analysis part), there may be several ways to design an inverse FB (the synthesis part) that reconstructs the signal after processing it [16]. However, the work presented here tackles the problem of signal classification after the analysis, without taking into account the inversion and even the invertibility of the FB.

TF analysis came up because of the Fourier transform inability to analyze time evolution. This is particularly useful for transient signals, which present different special frequency features according to the time. Nevertheless, the time may also be a source of intraclass variability if for instance the signals are not correctly aligned. Consequently, the next section details how to deal with that potential drawback.

2.2. Pooling functions

Signal classification usually needs some elaborate features based on local behaviors. This is mostly due to intraclass discrepancies, which often require nonlinearities to be compensated. This is the role of the pooling function ρ, we mentioned previously. The main pitfall the pooling function is aimed at overcoming is the random time shift inherent to real world signals acquisition. That is the pet hate shared by all the pooling functions used in this study (Table 2). Secondarily, pooling functions are also aimed at reducing the dimension of features in order to curb the curse of dimensionality.

Name	*Definition of* $\rho(\boldsymbol{x})$
ℓ_p-norm	$\left(\sum_{j=1}^{n} \lvert x(j)\rvert^p\right)^{\frac{1}{p}}$
Local-ℓ_p-norm	$\left(\left(\sum_{j=(i-1)w+1}^{iw} \lvert x(j)\rvert^p\right)^{\frac{1}{p}}\right)_{i=1}^{\lfloor \frac{n}{w} \rfloor}$
Max	$\left(\underset{(i-1)w+1\leq j\leq iw}{\texttt{max}}\ x(j)\right)_{i=1}^{\lfloor \frac{n}{w} \rfloor}$
Mean	$\left(\underset{(i-1)w+1\leq j\leq iw}{\texttt{mean}}\ x(j)\right)_{i=1}^{\lfloor \frac{n}{w} \rfloor}$
Scattering	$\lvert \boldsymbol{x}\rvert \star \boldsymbol{g}$

Table 2: Pooling functions. Parameters: $p \in \mathbb{N}$, $w \in \mathbb{N}$, $\boldsymbol{g}$ a Gaussian lowpass filter.

The most widespread pooling function may be the ℓ_p-norm (computed over the whole filtered signal), particularly with $p = 2$. This method is quite efficient against random time shifts and is thus often used in signal classification for the resulting information only depends on the frequency, not on the time evolution of the signal. However that lack of time resolution is also the main drawback of this first pooling function. A way to overcome it is to compute local invariant features, for instance by partitioning the input signal into a set of non-overlapping segments and, for each such sub-signal, outputting the maximum or

the mean value (Max and Mean pooling) These methods have been first introduced for CNNs [12] in order to absorb space shifts while keeping a trade-off with the space resolution. Finally, the scattering pooling, recently introduced by [24], has been designed for complex valued signals. The time shift invariance comes from both the modulus and the averaging operations. It is shown in [25] that this kind of pooling is linked with the logarithmic aggregation of the frequencies used to compute the Mel-Frequency Cepstral Coefficients (MFCCs).

In the Table 2, definitions are given for a single signal. Yet, these definitions are easily spread out to a set of signals (for instance a TF representation) through $\rho\colon \mathcal{X}^d \to \mathcal{X}^d$ defined by:

$$\rho\left(\{\boldsymbol{x_l}\}_{l=1}^d\right) \stackrel{\text{def}}{=} \{\rho(\boldsymbol{x_l})\}_{l=1}^d .$$

For instance, when applied to a TF representation of a signal, the ℓ_2-norm provides the marginal distribution of the energy for the given frequency bands.

The frequency features thus obtained (by filtering and pooling) can be discriminative if they are well designed. This brings out the central theme of the work presented in this paper: designing a TF transform that provides the classifier with the most discriminative transient features, in order to improve signal classification accuracy.

2.3. Filter bank learning

In the forthcoming sections, we introduce our novel framework for learning discriminative TF transforms in a large-margin setting. This framework comes along with several compelling properties that will be discussed next.

Our objective is to learn the features extracted from an FB, jointly with a decision function based on SVMs. For this purpose, let us consider a feature function $\rho \circ u\colon \mathcal{X} \to \mathcal{X}^d$, where $\rho\colon \mathcal{X}^d \to \mathcal{X}^d$ is a pooling function and u is a TF transform as defined in the equation (1). As explained earlier, the underlying assumption is that the features are extracted from the TF domain.

Given a training set $\{(\boldsymbol{x_i}, y_i)\}_{1\leq i\leq N}$ from $(\mathbb{K}^n \times \{-1,1\})^N$, a positive symmetric definite kernel $k\colon \mathcal{X}^d \times \mathcal{X}^d \to \mathbb{R}$ and a trade-off parameter C ($\frac{1}{N} \leq C$ [26]), designing a discriminative TF representation in a kernelized SVM framework can be formally formulated (like in [27, 3] for other purposes) by:

$$\underset{u\in\mathcal{T}}{\texttt{minimize}}\ J(u,k), \tag{2}$$

where $\mathcal{T}$ is the set of TF transforms of finite energy (see definitions in section 3) and $J(u,k)$ is the optimal value of the SVM problem depending on both the TF transform u and on the kernel k:

$$J(u,k) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} \underset{f\in\mathcal{H},b\in\mathbb{R},\boldsymbol{\xi}\in\mathbb{R}^N}{\texttt{min}}\ \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\mathbb{1}^T\boldsymbol{\xi} \\ \texttt{s.t.} \quad \left\{ \begin{array}{l} y_i\left(f((\rho\circ u)(\boldsymbol{x_i})) + b\right) \geq 1-\xi_i, \\ \forall i \in \mathbb{N}_N \\ \boldsymbol{\xi} \succcurlyeq 0, \end{array} \right. \end{array} \right. \tag{3}$$

where $\mathcal{H}$ is the reproducing kernel Hilbert space defined by the kernel function k [28] and $\boldsymbol{\xi}$ is the vector of slack variables. Note that the optimization program (3) is the standard problem solved by SVM algorithms. Moreover a pair (f^*, b^*) solution of (3) gives a learned decision function through:

$$\boldsymbol{x} \in \mathbb{K}^n \mapsto \operatorname{sign}\left(f^*((\rho\circ u)(\boldsymbol{x})) + b^*\right),$$

with f^* defined from a non-negative learned vector $\boldsymbol{\alpha^*}$ by:

$$\forall \boldsymbol{x} \in \mathbb{K}^n,\ f^*((\rho\circ u)(\boldsymbol{x})) \stackrel{\text{def}}{=} \sum_{\substack{i=1 \\ \alpha_i^*>0}}^{N} \alpha_i^* y_i k((\rho\circ u)(\boldsymbol{x_i}), (\rho\circ u)(\boldsymbol{x})). \tag{4}$$

Note that problem (2) of FB learning could have been formulated

$$\begin{array}{l} \underset{f\in\mathcal{H},b\in\mathbb{R},\boldsymbol{\xi}\in\mathbb{R}^N,u\in\mathcal{T}}{\texttt{minimize}}\ \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\mathbb{1}^T\boldsymbol{\xi} \\ \texttt{s.t.} \quad \left\{ \begin{array}{l} y_i\left(f((\rho\circ u)(\boldsymbol{x_i})) + b\right) \geq 1-\xi_i,\ \forall i \in \mathbb{N}_N \\ \boldsymbol{\xi} \succcurlyeq 0. \end{array} \right. \end{array} \tag{5}$$

Both problems (2) and (5) are hard in that they are non-convex. Indeed, as the filtered data is consecutively pooled and kernelized, each non-convexity in the pooling function ρ and in the mapping induced by the kernel k results in a non-convexity of the objective function with respect to the IRs [29, 3]. This effect can be accentuated if the IRs are naively parameterized in a non-convex way (for instance by the cutoff frequencies for a bandpass filter). As a consequence, the strategy of resolution of our learning problem is a major point to design an algorithm as efficient as possible, considering both classification accuracy and computation complexity. The reason to prefer the wrapper strategy (2) to the direct one (5) is that the former enables us to easily deal with infinite mappings induced by such kernels like the Gaussian one (definition (9)). Moreover, the wrapper strategy provides us with an algorithm that benefits from the the convexity of the SVM problem and from the current SVM solvers efficiency (for instance [30]) in terms of precision and of training time. Note that problems (2) and (5) are different but if they are solvable, then they have the same global minimas. As a consequence, choosing the problem (2) rather than (5) is equivalent to choosing a kind of resolution strategy.

3. Solving the problem

This section describes how to solve the FB learning problem. First, we exhibit the few hypotheses, that restrict the framework in which our work holds, and then we propose an algorithm to solve the resulting problem. The algorithm we present is an extension of the Generalized Multiple Kernel Learning algorithm [29] so as to handle a continuously parametrized family of kernels. Eventually, we detail how our framework interestingly enables to learn the pooling function, supposed fixed up to this point.

3.1. Problem restriction

It is possible to find a local minimum of (2) by performing a gradient descent directly on the IR coefficients (as for instance [3]) but this requires that all the operators are differentiable and may promote overfitting due to the high number of parameters. That is why, we propose another approach based on multiple kernels. To this end, we make three assumptions, respectively on the set of FBs, on the pooling functions and on the SVM kernels (which are the three consecutive stages of our processing line).

3.1.1. Filter banks

First, let us consider the set of FBs based on a finite number of normalized filters $\boldsymbol{h_\theta}$, picked from a continuously family parametrized by $\boldsymbol{\theta}$, and whose energies can be weighted with learned factors $\tilde{\mu}_{\boldsymbol{\theta}}$:

$$\mathcal{T} \stackrel{\text{def}}{=} \Big\{(\tilde{\mu}_{\boldsymbol{\theta}}\boldsymbol{h_\theta}, N_{\boldsymbol{\theta}})_{\boldsymbol{\theta}\in\mathcal{A}},\ \tilde{\boldsymbol{\mu}} \succcurlyeq 0, \|\tilde{\boldsymbol{\mu}}\|_{\ell_2} = 1 \text{ and } \mathcal{A} \subset \mathcal{P}\Big\},$$

where $\mathcal{P}$ is the continuous set of all possible IR parameters $\boldsymbol{\theta}$ and $\mathcal{A}$ is finite. We assume that the dimension of $\mathcal{P}$ is small, *i.e.* $\boldsymbol{\theta}$ is a vector with few components. Indeed, the idea behind choosing a family of filters is to control the complexity of the set $\mathcal{T}$ by decreasing the degree of freedom. This is a way to prevent overfitting.

The condition $\|\tilde{\boldsymbol{\mu}}\|_{\ell_2} = 1$ ensures that the FB is of unit energy. The main difficulty in this definition comes from the unknown number of filters used in the FB. Yet, let us reformulate $\mathcal{T}$ in a more convenient way:

$$\mathcal{T} = \Big\{(\tilde{\mu}_{\boldsymbol{\theta}}\boldsymbol{h_\theta}, N_{\boldsymbol{\theta}})_{\boldsymbol{\theta}\in\mathcal{P}},\ \tilde{\boldsymbol{\mu}} \succcurlyeq 0, \|\tilde{\boldsymbol{\mu}}\|_{\ell_2} = 1 \text{ and } \tilde{\boldsymbol{\mu}}\ FS\Big\},$$

FS (*Finite Support*) meaning that the number of non-zero values is finite. Consequently, even though an FB $(\tilde{\mu}_{\boldsymbol{\theta}}\boldsymbol{h_\theta}, N_{\boldsymbol{\theta}})_{\boldsymbol{\theta}\in\mathcal{P}}$ has theoretically an infinite amount of filters, only a finite number is actually active. The last definition of $\mathcal{T}$ highlights the fact that we will use the notion of sparsity to deal with the continuous set of parameters $\mathcal{P}$.

3.1.2. Pooling functions

Let u be a TF transform from $\mathcal{T}$ as defined in the previous subsection. Our next assumption is that the pooling function $\rho\colon \mathcal{X}^d \to \mathcal{X}^d$ is positive homogeneous of degree 1:

$$\forall \lambda \geq 0,\ \forall \boldsymbol{x} \in \mathbb{K}^d\colon\ \rho(\lambda u(\boldsymbol{x})) = \lambda(\rho \circ u)(\boldsymbol{x}).$$

This property is a very mild condition since it is verified by all the pooling functions presented in the Table 2.

Consider now the flattening operator $\operatorname{vec}(\cdot)$, that returns its input collapsed into a single dimension vector and let us compute the inner product between two pooled TF representations:

$$\begin{aligned}&\forall \boldsymbol{x}, \boldsymbol{z} \in \mathbb{K}^n,\ \langle \operatorname{vec}((\rho \circ u)(\boldsymbol{x})) \mid \operatorname{vec}((\rho \circ u)(\boldsymbol{z}))\rangle_{\ell_2}\\ &= \langle \operatorname{vec}(\rho(\{\downarrow N_{\boldsymbol{\theta}}[\tilde{\mu}_{\boldsymbol{\theta}}\boldsymbol{h_\theta} \star \boldsymbol{x}]\}_{\boldsymbol{\theta}\in\mathcal{A}})) \mid \operatorname{vec}(\rho(\{\downarrow N_{\boldsymbol{\theta}}[\tilde{\mu}_{\boldsymbol{\theta}}\boldsymbol{h_\theta} \star \boldsymbol{z}]\}_{\boldsymbol{\theta}\in\mathcal{A}}))\rangle_{\ell_2}\\ &= \textstyle\sum_{\boldsymbol{\theta}\in\mathcal{A}} \tilde{\mu}_{\boldsymbol{\theta}}^2 \langle \rho(\downarrow N_{\boldsymbol{\theta}}[\boldsymbol{h_\theta} \star \boldsymbol{x}]) \mid \rho(\downarrow N_{\boldsymbol{\theta}}[\boldsymbol{h_\theta} \star \boldsymbol{z}])\rangle_{\ell_2}.\end{aligned} \tag{6}$$

Roughly speaking, equation (6) means that, thanks to the positive homogeneous property, the inner product between two TF representations is a convex combination of the inner products between each frequency band (defined by a finite IR filter $\boldsymbol{h_\theta}$). The next subsection details why this relation is the key point of our approach.

3.1.3. SVM kernels

Let u be a TF transform from $\mathcal{T}$ and assume that the current set $\mathcal{A}$ is of size d. In this paper, we only consider two kinds of kernels $k\colon \mathcal{X}^d \times \mathcal{X}^d \to \mathbb{R}$: the linear and the Gaussian ones. If k is the linear kernel, then:

$$\begin{aligned}&\forall \overline{\boldsymbol{x}} \stackrel{\text{def}}{=} \{\boldsymbol{x_\theta}\}_{\boldsymbol{\theta}\in\mathcal{A}},\ \overline{\boldsymbol{z}} \stackrel{\text{def}}{=} \{\boldsymbol{z_\theta}\}_{\boldsymbol{\theta}\in\mathcal{A}} \in \mathcal{X}^d\colon\\ &k(\overline{\boldsymbol{x}}, \overline{\boldsymbol{z}}) \stackrel{\text{def}}{=} \langle \operatorname{vec}(\overline{\boldsymbol{x}}) \mid \operatorname{vec}(\overline{\boldsymbol{z}})\rangle_{\ell_2} = \textstyle\sum_{\boldsymbol{\theta}\in\mathcal{A}} \langle \boldsymbol{x_\theta} \mid \boldsymbol{z_\theta}\rangle_{\ell_2},\end{aligned} \tag{7}$$

where $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{z}}$ are two TF representations. By definition, the linear kernel is simply the inner product between the inputs [20]. If we now define the spanning kernels $(k_{\boldsymbol{\theta}})_{\boldsymbol{\theta}\in\mathcal{A}}$ and the multiple kernel $k_{[\boldsymbol{\mu}]}$ (where $\boldsymbol{\mu}$ is a vector of weights from $\mathbb{R}_+^d$ that will be learned) by:

$$k_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{z}}) \stackrel{\text{def}}{=} \langle \boldsymbol{x_\theta} \mid \boldsymbol{z_\theta}\rangle_{\ell_2} \quad \text{and} \quad k_{[\boldsymbol{\mu}]}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{z}}) \stackrel{\text{def}}{=} \sum_{\boldsymbol{\theta}\in\mathcal{A}} \mu_{\boldsymbol{\theta}} k_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{z}}), \tag{8}$$

then, by denoting $u_0 \stackrel{\text{def}}{=} \{(\boldsymbol{h_\theta}, N_{\boldsymbol{\theta}})\}_{\boldsymbol{\theta}\in\mathcal{A}}$ the FB of the normalized filters and by setting $\overline{\boldsymbol{x}} = (\rho \circ u)(\boldsymbol{x})$ along with $\overline{\boldsymbol{y}} = (\rho \circ u)(\boldsymbol{z})$, we get from (6) and (7):

$$\forall \boldsymbol{x}, \boldsymbol{z} \in \mathbb{K}^n,\ k((\rho \circ u)(\boldsymbol{x}), (\rho \circ u)(\boldsymbol{z})) = k_{[\tilde{\boldsymbol{\mu}}^2]}((\rho \circ u_0)(\boldsymbol{x}), (\rho \circ u_0)(\boldsymbol{z})),$$

where $\cdot^2$ is the pointwise square function. Basically, the similarity between two TF representations can be expressed thanks to the multiple kernel $k_{[\tilde{\boldsymbol{\mu}}^2]}$ as the convex combination of the similarities between each frequency band.

Respectively, if k is the Gaussian kernel of positive parameter γ, then:

$$\begin{aligned}&\forall \overline{\boldsymbol{x}} \stackrel{\text{def}}{=} \{\boldsymbol{x_\theta}\}_{\boldsymbol{\theta}\in\mathcal{A}},\ \overline{\boldsymbol{z}} \stackrel{\text{def}}{=} \{\boldsymbol{z_\theta}\}_{\boldsymbol{\theta}\in\mathcal{A}} \in \mathcal{X}^d\colon\\ &k(\overline{\boldsymbol{x}}, \overline{\boldsymbol{z}}) \stackrel{\text{def}}{=} \exp\left(-\gamma \| \operatorname{vec}(\overline{\boldsymbol{x}}) - \operatorname{vec}(\overline{\boldsymbol{z}}) \|_{\ell_2}^2\right)\\ &\qquad = \exp\left(-\gamma \textstyle\sum_{\boldsymbol{\theta}\in\mathcal{A}} \|\boldsymbol{x_\theta} - \boldsymbol{z_\theta}\|_{\ell_2}^2\right).\end{aligned} \tag{9}$$

The Gaussian kernel is a complex similarity function that reflects the proximity of its inputs mapped in an infinite-dimensional space [20]. If we define this time the spanning kernels $(k_{\boldsymbol{\theta}})_{\boldsymbol{\theta}\in\mathcal{A}}$ and the multiple kernel $k_{[\boldsymbol{\mu}]}$ by:

$$\begin{aligned}&k_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{z}}) \stackrel{\text{def}}{=} \exp\left(-\gamma \|\boldsymbol{x_\theta} - \boldsymbol{z_\theta}\|_{\ell_2}^2\right) \quad \text{and}\\ &k_{[\boldsymbol{\mu}]}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{z}}) \stackrel{\text{def}}{=} \textstyle\prod_{\boldsymbol{\theta}\in\mathcal{A}} \left(k_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{z}})\right)^{\mu_{\boldsymbol{\theta}}},\end{aligned} \tag{10}$$

then from (6) and (9), we get:

$$\forall \boldsymbol{x}, \boldsymbol{z} \in \mathbb{K}^n,\ k((\rho \circ u)(\boldsymbol{x}), (\rho \circ u)(\boldsymbol{z})) = k_{[\tilde{\boldsymbol{\mu}}^2]}((\rho \circ u_0)(\boldsymbol{x}), (\rho \circ u_0)(\boldsymbol{z})).$$

The last relation means that comparing two TF representations with a Gaussian kernel is like computing a multiplicative combination of the similarities between each frequency band in an infinite-dimensional space.

Thanks to (6), we have shown that for both kernels k (linear and Gaussian), the problem of learning the weights $\tilde{\mu}_{\boldsymbol{\theta}}$ of an FB boils down to learning some parameters $\mu_{\boldsymbol{\theta}} \overset{\text{def}}{=} \tilde{\mu}_{\boldsymbol{\theta}}^2$ of the SVM kernel $k_{[\boldsymbol{\mu}]}$, called a multiple kernel [27].

Concerning the choice of a kernel, we recommend to use the Gaussian kernel (which is in practice rarely worse than the linear one) expect if the practitioner does not have enough spare time to perform the cross-validation on both the cost parameter C and the kernel parameter γ. In this case, we recommend to choose the linear kernel, which is quicker to compute and to cross-validate (the only parameter to determinate is C).

3.2. Learning the TF transform

With the keys given in the previous sections, the problem of learning a TF transform (2) becomes the one of learning an infinite combination of kernels:

$$\underset{\boldsymbol{\mu}\in\mathbb{R}^{\mathcal{P}}}{\texttt{minimize}}\ J\left(u_0, k_{[\boldsymbol{\mu}]}\right) \quad \texttt{s.t.} \quad \begin{cases} \mathbb{1}^T\boldsymbol{\mu} = 1 \\ \boldsymbol{\mu} \succcurlyeq 0 \\ \boldsymbol{\mu}\ FS, \end{cases} \tag{11}$$

where $u_0 \overset{\text{def}}{=} (\boldsymbol{h}_{\boldsymbol{\theta}}, N_{\boldsymbol{\theta}})_{\boldsymbol{\theta}\in\mathcal{P}}$. When the set $\mathcal{P}$ is finite, this kind of problem can be solved thanks to existing Multiple Kernel Learning solvers like the convex one from [31] for the linear kernel (7) and a variant of the non-convex one from [29] for the Gaussian kernel (9) [1]. Nevertheless, in our case $\mathcal{P}$ is infinite due to the continuous nature of the filter parameter $\boldsymbol{\theta}$. Thus, one of the main contributions of this paper is to propose Algorithm 1 to solve problem (11).

In practice, such an algorithm already exists when k is the linear kernel (7) thanks to the so-called Infinite Kernel Learning approach (IKL) [32]. IKL is a problem introduced and solved by Gehler and Nowozin [32] to spread out linear MKL to infinitely many kernels. The learning problem is turned into a dual Semi-Infinite Linear Program, calling upon the strong duality of the problem (it is convex). Then, it is solved with a delayed constraint generation algorithm.

The algorithm proposed in this paper extends the state of the art by being the only one to provide a solution to the problem of FB learning (11) when the kernel is Gaussian. Concretely, our algorithm turns out to be an extension of the one from [29] so as to handle a continuously parametrized family of kernels $(k_{\boldsymbol{\theta}})_{\boldsymbol{\theta}\in\mathcal{P}}$ as defined in (10).

[1] When the kernel is Gaussian, the problem studied here is slightly different from the one originally introduced in [29] since we replaced the Tikhonov regularization on $\boldsymbol{\mu}$ by an Ivanov one. This replacement naturally comes from our hypotheses but there are two other reasons for using an explicit constraint on $\boldsymbol{\mu}$: first, there is no regularization coefficient to tune (which is hard in practice for it requires either a deep theoretical study or a lot of computational resources) and then, since $\boldsymbol{\mu}$ is guaranteed to be on the unit sphere of the ℓ_1-norm, the cost parameter C carries on its original role.

Data: training dataset $\{(\boldsymbol{x_i}, y_i)\}_{1\leq i\leq N}$.
Result: (sub)optimal filter bank u^* and optimal classifier f^*.

1 $\mathcal{A} \leftarrow$ linear grid of IR parameters;
2 $\bar{\boldsymbol{\mu}} \leftarrow \frac{1}{\text{card}(\mathcal{A})}\mathbb{1}$ {*initial weights*};
3 **while** *not suboptimal* **do**
4 $u \leftarrow (\boldsymbol{h}_{\boldsymbol{\theta}}, N_{\boldsymbol{\theta}})_{\boldsymbol{\theta}\in\mathcal{A}}$ {*bank of normalized filters*};
5 $(\boldsymbol{\mu^*}, f^*) \leftarrow$ Solve MKL with $(u, (k_{\boldsymbol{\theta}})_{\boldsymbol{\theta}\in\mathcal{A}})$, initialized with $\bar{\boldsymbol{\mu}}$ {*formulas* (8) *or* (10)};
6 $\mathcal{A} \leftarrow \{\boldsymbol{\theta} \in \mathcal{A}, \mu^*_{\boldsymbol{\theta}} > 0\}$;
7 $\Theta \leftarrow$ random sample from $\mathcal{P}$;
8 $\hat{\boldsymbol{\theta}} \leftarrow \underset{\boldsymbol{\theta}\in\Theta}{\text{argmax}}\ V(\boldsymbol{\theta})$;
9 **if** $V(\hat{\boldsymbol{\theta}}) > \sum_{\boldsymbol{\theta}\in\mathcal{A}} \mu_{\boldsymbol{\theta}} V(\boldsymbol{\theta})$ **then** {*optimality condition violated*}
10 $\mathcal{A} \leftarrow \mathcal{A} \cup \{\hat{\boldsymbol{\theta}}\}$;
11 $\bar{\boldsymbol{\mu}} \leftarrow [\boldsymbol{\mu}_{\mathcal{A}}; 0]$;
12 **else**
13 Suboptimality reached;
14 $u^* \leftarrow (\sqrt{\mu^*_{\boldsymbol{\theta}}}\boldsymbol{h}_{\boldsymbol{\theta}}, N_{\boldsymbol{\theta}})_{\boldsymbol{\theta}\in\mathcal{A}}$;

Algorithm 1: Filter-MKL algorithm.

Our algorithm also applies to the case of FB learning (11) with a linear kernel (the differences with IKL are discussed in section 6).

The proposed approach tackles the problem in its primal form. It is based on the so-called active set principle [33] and has the flavor of [6]: let us start with a guess $\mathcal{A}$ ($\mathcal{A}$ is a finite set of parameters, verifying $\mathcal{A} \subseteq \mathcal{P}$) on the set of solution parameters $\mathcal{P}^*$ (suppose that an oracle gave it to us) and then solve the multiple kernel problem with respect to $(\mu_{\boldsymbol{\theta}})_{\boldsymbol{\theta}\in\mathcal{A}}$ (line 5 in Algorithm 1). For the linear kernel, we use the MKL solver from [31] while for the Gaussian one the optimization strategy used here is a reduced gradient descent [34] with a backtracking linesearch. This step can be seen as performing of block-coordinate descent considering that $\mu_{\boldsymbol{\theta}}$ vanishes for $\boldsymbol{\theta}$ in $\mathcal{P}\backslash\mathcal{A}$. It results in an active set $\mathcal{A}^*$ of parameters whose weights are non-zeros and in its complementary non-active set $\mathcal{A}\backslash\mathcal{A}^*$. If $\mathcal{A}$ includes $\mathcal{P}^*$ then the optimality conditions are verified for all $\boldsymbol{\theta}$ in $\mathcal{P}$. By contraposition, if the optimality conditions are not verified for a $\boldsymbol{\theta}$ in $\mathcal{P}$, then $\mathcal{A}$ does not include $\mathcal{P}^*$ and specially, the violator $\boldsymbol{\theta}$ is missing from the guess $\mathcal{A}$. So let us update $\mathcal{A}$ with the rule $\mathcal{A} \leftarrow \mathcal{A}^* \cup \{\boldsymbol{\theta}\}$ and solve again the multiple kernel problem. By doing iteratively these two steps, the algorithm performs a descent on an infinite amount of parameters.

We now address how to check the optimality of a given weighting vector $\boldsymbol{\mu}$ with respect to problem (11) (line 9 in Algorithm 1). If the optimality is not verified by a weight $\mu_{\boldsymbol{\theta}}$, then we have to add the parameter $\boldsymbol{\theta}$ to the current set $\mathcal{A}$ and iterate. The way to find such a $\boldsymbol{\theta}$ is discussed in section 4, so we only focus on the optimality condition. Let $\boldsymbol{Y} \overset{\text{def}}{=} \text{diag}(\boldsymbol{y})$ be the matrix of labels, $\boldsymbol{D}_{\boldsymbol{\theta}}$ and $\boldsymbol{K}_{[\boldsymbol{\mu}]_+}$

respectively the matrix of distances in the TF space and the kernel matrix of data:

$$\boldsymbol{D_\theta} \overset{\text{def}}{=} \left(\|\rho\left(\downarrow N_{\boldsymbol{\theta}}\left[\boldsymbol{h_\theta} \star \boldsymbol{x_i}\right]\right) - \rho\left(\downarrow N_{\boldsymbol{\theta}}\left[\boldsymbol{h_\theta} \star \boldsymbol{x_j}\right]\right)\|_{\ell_2}^2\right)_{1\leq i,j\leq N},$$
$$\boldsymbol{K}_{[\boldsymbol{\mu}]_+} \overset{\text{def}}{=} \left(k_{[\boldsymbol{\mu}]}((\rho\circ u_0)(\boldsymbol{x_i}),(\rho\circ u_0)(\boldsymbol{x_j}))\right)_{1\leq i,j\leq N}.$$

As detailed in 7, the main equilibrium condition to be violated in our algorithm when the kernel k is Gaussian is:

$$\forall \boldsymbol{\theta}\in\mathcal{P},\ V(\boldsymbol{\theta}) \leq \sum_{\substack{\boldsymbol{\theta'}\in\mathcal{P},\\ \mu_{\boldsymbol{\theta'}}>0}} \mu_{\boldsymbol{\theta'}} V(\boldsymbol{\theta'}),$$

where

$$V(\boldsymbol{\theta}) = -{\boldsymbol{\alpha^*}}^T \boldsymbol{Y}\left(\boldsymbol{D_\theta}\circ \boldsymbol{K}_{[\boldsymbol{\mu}]_+}\right)\boldsymbol{Y}\boldsymbol{\alpha^*},$$

and $\boldsymbol{\alpha^*}$ is the optimal dual variable of the SVM problem applied to $\{(\rho\circ u_0)(\boldsymbol{x_i}),y_i)\}_{1\leq i\leq N}$ with the kernel $k_{[\boldsymbol{\mu}]}$.

Respectively, when the kernel k is linear, the main condition to be violated is identical with [32, 6]:

$$V(\boldsymbol{\theta}) = {\boldsymbol{\alpha^*}}^T \boldsymbol{Y}\boldsymbol{K}_{\boldsymbol{\theta}+}\boldsymbol{Y}\boldsymbol{\alpha^*},$$

where $\boldsymbol{K}_{\boldsymbol{\theta}+}$ is a spanning kernel matrix defined by:

$$\boldsymbol{K}_{\boldsymbol{\theta}+} \overset{\text{def}}{=} \left(k_{\boldsymbol{\theta}}((\rho\circ u_0)(\boldsymbol{x_i}),(\rho\circ u_0)(\boldsymbol{x_j}))\right)_{1\leq i,j\leq N}.$$

In both cases (linear and Gaussian kernels), the optimal decision function resulting from solving the learning problem is formulated by:

$$\begin{array}{rl} \forall \boldsymbol{x}\in\mathbb{K}^n, f^*\left((\rho\circ u^*)(\boldsymbol{x})\right) & \\ \overset{\text{def}}{=} & \sum_{i=1}^N \alpha_i^* y_i k_{[\boldsymbol{\mu^*}]}\left((\rho\circ u_0)(\boldsymbol{x}),(\rho\circ u_0)(\boldsymbol{x_i})\right) \\ = & \sum_{i=1}^N \alpha_i^* y_i k\left((\rho\circ u^*)(\boldsymbol{x}),(\rho\circ u^*)(\boldsymbol{x_i})\right), \end{array} \tag{12}$$

where $\boldsymbol{\alpha^*}$ is the optimal dual variable of the SVM problem applied to $\{(\rho\circ u_0)(\boldsymbol{x_i}),y_i)\}_{1\leq i\leq N}$ with the kernel $k_{[\boldsymbol{\mu^*}]}$, $\boldsymbol{\mu^*}$ is the optimal vector of weights from the MKL problem and $u^* = (\sqrt{\mu_{\boldsymbol{\theta}}^*}\boldsymbol{h_\theta}, N_{\boldsymbol{\theta}})_{\boldsymbol{\theta}\in\mathcal{A}}$. Note that when the optimal TF transform u^* is known, the decision function from (12) turns out to be the one from a single SVM with a usual kernel k.

3.3. Learning the pooling function

Previous works [35, 36] show the importance of choosing an appropriate pooling function, as it is part of the very first stages of the processing line. Following this observation, an attempt at designing data-driven pooling functions recently appeared [37]. Interestingly, the framework we built in this article, also enables to learn the pooling function as the concatenation of several pooling functions. This kind of functions, that we call *multiple pooling function*, is defined as:

$$\mathcal{F} \overset{\text{def}}{=} \left\{(\tilde{\eta}_r\rho_r)_{1\leq r\leq p},\ \tilde{\boldsymbol{\eta}} \succcurlyeq 0 \text{ and } \|\tilde{\boldsymbol{\eta}}\|_{\ell_2} = 1\right\},$$

where p is the positive number of spanning pooling functions ρ_r and $\boldsymbol{\eta}$ is the weighting vector. Then, for any multiple pooling function ρ from $\mathcal{F}$,

$$\begin{array}{l} \forall \boldsymbol{x},\boldsymbol{z}\in\mathbb{K}^n, \langle(\rho\circ u)(\boldsymbol{x}) \mid (\rho\circ u)(\boldsymbol{z})\rangle_{\ell_2} \\ = \sum_{r=1}^p \tilde{\eta}_r^2 \langle(\rho_r\circ u)(\boldsymbol{x}) \mid (\rho_r\circ u)(\boldsymbol{z})\rangle_{\ell_2}. \end{array} \tag{13}$$

This relation is a mirror image of (6) for multiple pooling functions. Consequently, learning the pooling function can be seen as learning a multiple kernel in the same way as it has been done before to learn an FB. In this context, given a TF transform u, the optimal decision function resulting from solving the problem of learning the pooling function is formulated by:

$$\begin{array}{rl} \forall \boldsymbol{x}\in\mathbb{K}^n, f^*\left((\rho^*\circ u)(\boldsymbol{x})\right) & \\ \overset{\text{def}}{=} & \sum_{i=1}^N \alpha_i^* y_i k_{[\boldsymbol{\eta^*}]}\left((\rho_0\circ u)(\boldsymbol{x}),(\rho_0\circ u)(\boldsymbol{x_i})\right) \\ = & \sum_{i=1}^N \alpha_i^* y_i k\left((\rho^*\circ u)(\boldsymbol{x}),(\rho^*\circ u)(\boldsymbol{x_i})\right), \end{array} \tag{14}$$

where $\rho_0 \overset{\text{def}}{=} (\rho_r)_{1\leq r\leq p}$, $\boldsymbol{\alpha^*}$ is the optimal dual variable of the SVM problem applied to $\{(\rho_0\circ u)(\boldsymbol{x_i}),y_i)\}_{1\leq i\leq N}$ with the kernel $k_{[\boldsymbol{\eta^*}]}$, $\boldsymbol{\eta^*}$ is the optimal vector of weights from the MKL problem and $\rho^* = (\sqrt{\eta_r^*}\rho_r)_{1\leq r\leq p}$.

Learning the pooling function seems advantageous, since the practitioner does not need to choose a particular pooling function beforehand (in practice, it is quite difficult to have an intuition on which pooling function will perform the best), nevertheless we have to be careful since adding new variables to the optimization scheme may promote overfitting. For this purpose, we propose a three-stage approach: in a first step, we draw a linear grid of parameters for the finite IR filters and we learn the pooling function with this first FB. Then, we solve (11) (Algorithm 1) with the learned pooling function and finally learn it again with the optimal FB. This is the approach used in the numerical experiments (section 5). Unreported experiments show that this approach is more efficient than including the pooling learning in each iteration of Algorithm 1.

4. Discussions

We now discuss two points of the proposed algorithm. The first one is the way to concretely check the optimality condition (that is how to find a violator for the active set strategy and how to define a stopping criterion). The second one deals with the convergence of the algorithm and with the normalization of the kernels.

4.1. Column generation

There are two difficulties in checking the optimality conditions of a problem with an infinite amount of parameters. The first one is to find parameters that violate the equilibrium conditions, in order to run the active set strategy. As in [38], this can be done by randomization (line

7 in Algorithm 1). The second difficulty is to solve the variational subproblem:

$$\underset{\boldsymbol{\theta}\in\mathcal{P}}{\texttt{maximize}}\; V(\boldsymbol{\theta}), \tag{15}$$

in order to stop the process. Indeed, if a maximizer $\hat{\boldsymbol{\theta}}$ satisfies $V(\hat{\boldsymbol{\theta}}) \leq \sum_{\substack{\boldsymbol{\theta}\in\mathcal{P},\\ \mu_{\boldsymbol{\theta}}>0}} \mu_{\boldsymbol{\theta}} V(\boldsymbol{\theta})$, then there is no more violator and the system is at an equilibrium.

In practice, this problem is very hard to solve (because of its nonconvexity) and it is accepted that if no parameter drawn by randomization at a given iteration violates the equilibrium condition, then this is true for all parameters [38] and the algorithm can stop.

In [32], a similar subproblem is solved thanks to a Newton method initialized with several points. On the contrary, the main point of using randomization here is a complexity concern: gradient-based techniques are generally slow because of the gradient computation and of the curvature of the objective, and even more when the process is repeated with several initializations. Moreover as the randomization approach does not compute any gradient, it can address non-smooth functions with respect to $\boldsymbol{\theta}$ (for instance when using a Max pooling function).

Akin to [32], which initializes the gradient-based algorithm with violator parameters from previous iterations, the randomization step can be driven thanks to a probability distribution based on the knowledge acquired at the previous iteration. Suppose that, at the first iteration, spanning kernels are built on a uniform grid of parameters, that spans the set $\mathcal{P}$. Then, the solution of the multiple kernel problem gives a rough guess of the discriminative power distribution over $\mathcal{P}$, thanks to the objective function V of (15). Each iteration is thus aimed at refining the previous solution more than at discovering new ones.

An estimation of this distribution is directly proportional to $\boldsymbol{\theta} \in \mathcal{P} \mapsto \texttt{max}\ \left(0, V(\boldsymbol{\theta}) - \sum_{\boldsymbol{\theta}\in\mathcal{A}} \mu_{\boldsymbol{\theta}} V(\boldsymbol{\theta})\right)$. One option to refine the sampling is then to regress the (normalized) previous function and to sample some filter parameters following this regression, using for instance a Metropolis-Hastings algorithm. In practice, thousands of realizations are needed to come close to the regressed distribution while in the proposed algorithm only few hundreds are sampled. As a consequence, in practice our algorithm randomizes some parameters following both a uniform distribution over $\mathcal{P}$ and a uniform distribution over a small box centered on the violator parameter of the previous iteration. Again, this heuristics is more aimed at refining the selected parameters than at speeding up the algorithm.

4.2. Computational considerations

Due to the non-convexity of our learning problem, we do not look for a global minimum. Theoretically, the algorithm may stop on a local maximum, but in practice this is very unlikely, since such an equilibrium is unstable. We are yet ensured that the objective value strictly improves at each step. This is so since at each iteration, we build upon a multiple kernel, another one which is exactly the same except for a new kernel with a null weight. This new multiple kernel is not a critical point since the added kernel violates the equilibrium conditions. Consequently, solving a multiple kernel problem with this new set of kernels from the given set of weights ensures to make a steady progress.

Learning a multiple kernel supposes to compare discriminative power of kernels. For this reason, kernels must be approximately of the same *magnitude*, otherwise some kernels may get a major role in the multiple kernel only because of their high *magnitude* but without being discriminative. This is one of the pitfalls around minimizing the SVM objective and we have to be careful with it. To prevent this effect, kernels are set up with the following rule: if it is a linear kernel, it is normalized by its trace; if it is a Gaussian kernel, the distance matrix is normalized by its Frobenius norm. In both cases, the normalization factor is propagated to the learned weighting vectors $\boldsymbol{\mu}^*$ and $\boldsymbol{\eta}^*$ when creating the optimal FB and the optimal pooling function.

5. Numerical experiments

This section is aimed at demonstrating the appeal of the proposed method with several experiments. First of all, we give some examples of IR parametrization for which parameters can be learned by our algorithm. Then, we deal with a hand-crafted problem to describe a basic application. In addition, we tackle two real-world situations: a Brain Computer Interface classification problem and a scene classification task.

5.1. Settings

The first simple IR parametrization which can be considered is a bandpass filter, designed through a window method. Let us note ω_{on} and ω_{off} the normalized cut-off frequencies of the filter ($0 \leq \omega_{\text{on}}, \omega_{\text{off}} \leq \pi$). Then $\boldsymbol{\theta} \stackrel{\text{def}}{=} (\omega_{\text{on}}, \omega_{\text{off}})$ and $\mathcal{P} \stackrel{\text{def}}{=} [0,\pi]^2$. The IR (of length q) is therefore:

$$\forall (\omega_{\text{on}}, \omega_{\text{off}}) \in \mathcal{P},\ \forall t \in \mathbb{N}_q :$$
$$\boldsymbol{h}_{(\boldsymbol{\omega}_{\text{on}}, \boldsymbol{\omega}_{\text{off}})}(t) \stackrel{\text{def}}{=} \frac{\sin(\omega_{\text{off}} t - \omega_{\text{off}} \frac{q}{2}) - \sin(\omega_{\text{on}} t - \omega_{\text{on}} \frac{q}{2})}{t} \frac{W(t)}{\nu},$$

where W is a window function (for instance Hanning or Blackman) and ν is a normalization factor to get a unitary energy.

Another option is the Morlet wavelet. It consists in a complex-valued wave modulated by a Gaussian window of width σ. The envelope factor σ controls the number of oscillations in the wave packet. The parameter vector is $\boldsymbol{\theta} \stackrel{\text{def}}{=} (\tau, \sigma)$ and lives in $\mathcal{P} \stackrel{\text{def}}{=} [1, \beta_\tau] \times [1, \beta_\sigma]$ (where β_τ and β_σ are upper bounds, for instance 50). The IR formula is then:

$$\forall(\tau,\sigma)\in\mathcal{P},\ \forall t\in\mathbb{N}_q\text{:}$$
$$\boldsymbol{h}_{(\boldsymbol{\tau},\boldsymbol{\sigma})}(t)\stackrel{\text{def}}{=}\mathrm{e}^{-8\left(\frac{\pi\tau(t-\frac{q}{2})}{\sigma q}\right)^2}\left(\mathrm{e}^{4\imath\pi\frac{\tau(t-\frac{q}{2})}{n}}-\mathrm{e}^{-\sigma^2/2}\right)\frac{1}{\nu},$$

where ν is still a normalization factor. The complex-valued wavelet approach in union with the scattering pooling is quite interesting since, as the Morlet wavelet is almost analytic for $\sigma > 5$ (*i.e.* its Fourier transform is almost null for negative frequencies), it is close to a one-stage scattering transform, which has been proved to be efficient for signal classification [25].

Finally, the methods we will confront in this section are summarized in the Table 3. The bulk of them ends with an SVM classifier. The method, nicknamed *SVM* in the table and the figures, is a naive approach considering the time series as the features. *Max* builds upon *SVM* and pools the signals by computing local maxima. On the contrary to both last methods, *DFT* considers the magnitude of the discrete Fourier transformations of the signals and *MFCC* their MFCCs. Those four methods are the baselines for our study.

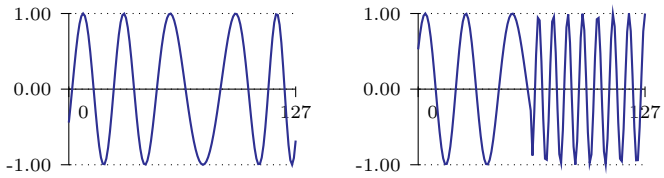
We also consider advanced methods like Convolutional Neural Networks [12] (*CNN*) and Wavelet Kernel Learning [6] (*WKL*). All the study long, a CNN has one convolutional layer with three feature maps. The shrinkage factor of the subsampling layer is the same as the one in the Max and Mean pooling used with the methods proposed in this paper ($N_{\boldsymbol{\theta}}$ being set to 1 for all $\boldsymbol{\theta}$). These ones are called *Band-Max*, *Band-*ℓ_2 and *Morlet-Scattering*, according to the family of filters (bandpass or Morlet wavelet) and to the kind of pooling function (see Table 2).

For the real datasets (BCI and scene classification), our method includes both without and with learning the pooling function (see Section 3.3). In this last case, our method is nicknamed *Band-Pooling* or *Morlet-Pooling*, according to the family of filters, and the multiple pooling function is made up of Max, Mean and Scattering pooling with different values of window w (see Table 2) and of ℓ_p-norms ($p \in \{1, 2\}$). Moreover, the weights of the pooling functions are uniformly initialized.

The results of classification accuracy given in this section are the ones obtained with the test dataset and with a Gaussian SVM kernel, since those with a linear one are not better. For the purpose of the study, these results are given through statistics based on 10 runs for each of which the dataset is randomly split into non-overlapping training and test sets (with a ratio of 70-30%). Moreover the SVM parameters C and γ are tuned through a five-fold validation resampling of the training set. Finally, before each learning step, the training set is normalized to unit magnitude, and the test set is rescaled accordingly.

5.2. Toy dataset

The considered toy dataset is made up of two classes based on different patterns (Figure 3). The first pattern is a sine curve with a normalized frequency that varies around 0.039. The second one is the same on the first half and then, is formed by another sine curve with a higher normalized frequency that varies around 0.117. Variable frequencies are the first kind of intraclass distortions that characterize our toy dataset. The second kind of intraclass discrepancies introduced is a slight random time-shift. In the end, a random colored noise is added to each signal. This one comes from a Gaussian white noise that is filtered by a filter randomly chosen among $[1, -2, 1]$, $[0, 1, -1]$ and $[1, 0, 1]$. This noise is stationary at the scale of the signal but nonstationary at the scale of the dataset (considering that the dataset comes from the segmentation of a single long time signal).



(a) Class 1. (b) Class 2.

Figure 3: Toy dataset.

Obviously, the dataset has been designed in order to invalidate other approaches (Table 3). Indeed, Figures 4 and 5 show that the Fourier Transform (*DFT*) is not suitable for this problem and that the Shannon representation (*SVM*) has difficulties to handle the colored noise. Yet the use of a Max pooling function jointly with the Shannon representation (*Max*) improves the classification accuracy. The Convolutional Neural Network (*CNN*), as for it, tends to overfit with small training datasets and with a low SNR. The Wavelet Kernel Learning approach (*WKL*) performs well but does not seem to handle really noisy data. In comparison to all those methods, the proposed approach applied with bandpass filters (*Band-Max*) is always on top thanks to the filters learning in union with a Max pooling function.

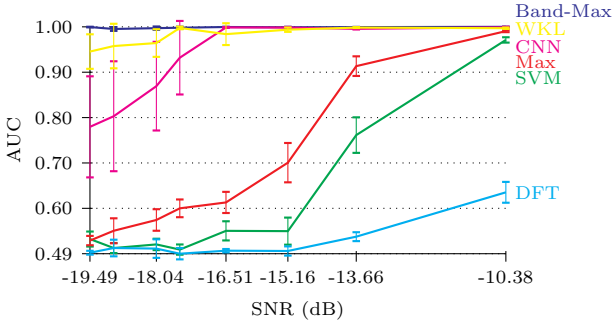


Figure 4: Toy dataset: accuracy with respect to the SNR.

Figure 6 shows an instance of learned FBs. The main discriminative filter is a bandpass one centered on the nor-

Abbreviation	*Analysis*	*Pooling*	*Classifier*	*Remark*
SVM	-	-	SVM	
Max	-	Max	SVM	
DFT	Discrete Fourier Transform	-	SVM	
MFCC	Mel-Frequency Cepstrum	-	SVM	
CNN	Convolutional Neural Network	Mean	MLP	[12]
WKL	Wavelets	ℓ_2-norm	MKL	[6]
Band-Max	Learned bank of bandpass filters	Max	SVM	Proposed method
Band-ℓ_2	Learned bank of bandpass filters	ℓ_2-norm	SVM	Proposed method
Morlet-Scattering	Learned bank of Morlet wavelets	Scattering	SVM	Proposed method
Band-Pooling	Learned bank of bandpass filters	Learned	SVM	Proposed method
Morlet-Pooling	Learned bank of Morlet wavelets	Learned	SVM	Proposed method

Table 3: Confronted methods.

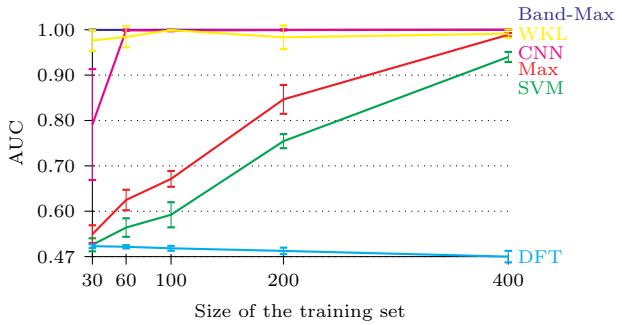


Figure 5: Toy dataset: accuracy with respect to the size of the training set.

malized frequency 0.117. This is indeed the most discriminative feature between the two classes of signals.

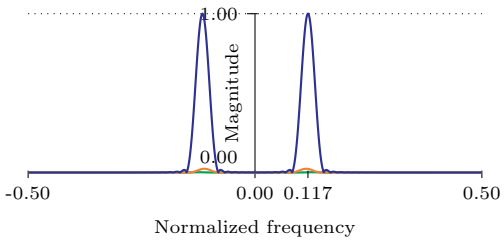


Figure 6: Toy dataset: instance of learned filter banks (each color embodies a filter in the Fourier domain).

5.3. Real-world problems

BCI problem. Akin to [6], we consider as a first real-world case, a Brain Computer Interface problem. Electroencephalographic activity has been recorded from a healthy man with a cap containing 32 tin electrodes placed on standard positions. The subject performed imaginary plantar flexions of the right foot. The aim is to reach a given target torque, either as fast as possible or following a 4s linear torque increase. Both protocols (quick and slow flexions) generate different Movement-Related Cortical Potentials (MRCPs) that we want to classify. Thus, it is a biclass problem consisting in single trial classification of MRCPs. The dataset contains 75 signals (of length 512) from each class, as many as imaginary tasks performed by the subject. Experiments have been undertaken on the channels 9, 12, 17, 29 and 30 as in [6].

Acoustic scenes. As a second real-world case, we tackle a Computational Auditory Scene Analysis (CASA) situation. CASA is the field of computer science that looks for how to mimic the ability of Humans to follow specific sound sources in a complex audio environment [39]. Two kinds of problems fall in CASA: acoustic event detection and acoustic scene classification. As for us, we are only interested in the problem of acoustic scenes (or soundscapes) classification. Its aim is to characterize the environment of an audio stream by providing a semantic label. For instance, the database we are interested in is made up of 10 classes [39]: busy street, quiet street, park, open-air market, bus, subway-train (or tube), restaurant, store / supermarket, office and subway station (or tube station). The database contains 10 audio recordings (30-second segments) for each scene. To make it, three different recordists recorded each scene type. They visited a wide variety of locations in Greater London over a period of months (Summer and Autumn 2012), with moderate weather conditions and varying times of day and week.

In our experiment, each recording is split into 300-millisecond frames, filtered and downsampled in order to reduce the computational complexity. The goal is then to label these subsignals in the framework of several binary classification problems, for instance tube *vs* tube station. The baseline we consider is an SVM classifier applied to the MFCCs (*MFCC*). This approach, which is often used to classify audio signals, seems really efficient on some of the biclass problems studied in this paper (for instance office *vs* supermarket).

Results. The box plots in Figure 7 (BCI experiment) bring out that the naive approaches, considering either the signals as classification features (*SVM*) or the magnitudes of the Fourier transform (*DFT*), fail to correctly generalize from the training dataset. This proves the appeal of data-driven methods for signal classification. Moreover, it

seems that a Morlet wavelet along with a Scattering pooling (*Morlet-Scattering*) achieves comparable but slightly better results than a bandpass FB with a Max pooling (*Band-Max*), and really better results than bandpass filters with an ℓ_2-norm (*Band-ℓ_2*). As the latter method corresponds to considering the marginal distribution of the energy for discriminative frequencies (*i.e.* there is no information in time), it demonstrates the improvement of TF methods compared to frequency approaches for this numerical experiment. Nevertheless, for the scenes dataset (Figures 8), this variant often performs the best. This is so probably because subsignals are quite short.

For both experiments, there is always a variant of our method that outperforms the other competitive approaches. For *WKL*, we used the same setting as in the paper that introduced the method [6] (full stochastic WKL with a 6-tap wavelet and a marginal Gaussian kernel, which demonstrated better results than without pooling the filtered signals) and recovered similar results than in the latter paper for the BCI experiment, even though the signals have not been normalized in the same way (zero mean and unit variance in [6] and unit magnitude in this paper). Some insights of our preeminence are that i) our method enables several families of filters and several ways of pooling, ii) the parameter of the Gaussian kernel is fixed in *WKL* while it is not in our method, iii) our approach with a Gaussian kernel naturally needs a non-linear multiple kernel, which as been proved to be better than a linear one for some problems [29], while *WKL* systematically uses a linear multiple kernel.

Our approach looks also truly better than *CNN*. Indeed *CNN* shows poor results, most likely because of the high level of noise in the BCI signals and more generally speaking because of the difficulty to correctly train them without being a specialist. On the contrary, our method is easy to tune for real-world situations.

With minor exceptions, learning the pooling function (methods *Band-Pooling* and *Morlet-Pooling*) gives quite similar results (and even better for several problems) to the best pooling function chosen by hand. This leads us to an interesting statement: the practitioner does not need to have an intuition on which pooling function to use, or to try many of them independently. Learning the pooling function enables to automatically tune our method and generally gives results comparable to a handcrafted approach.

6. Related works

6.1. Convolutional Neural Networks

Considering an FB to model a discriminative TF transform leads us to an already addressed problem: the Convolutional Neural Networks [12]. CNNs are the state of the art in many pattern recognition problems, like character recognition. The efficient machinery of a CNN is built upon one or several stages of FBs that filter the signals and reduce their dimension. Each FB contains a nonlinearity thanks to a so-called activation function. Then, the resulting features feed a Multi-Layer Perceptron (MLP), which is a non-linear classifier.

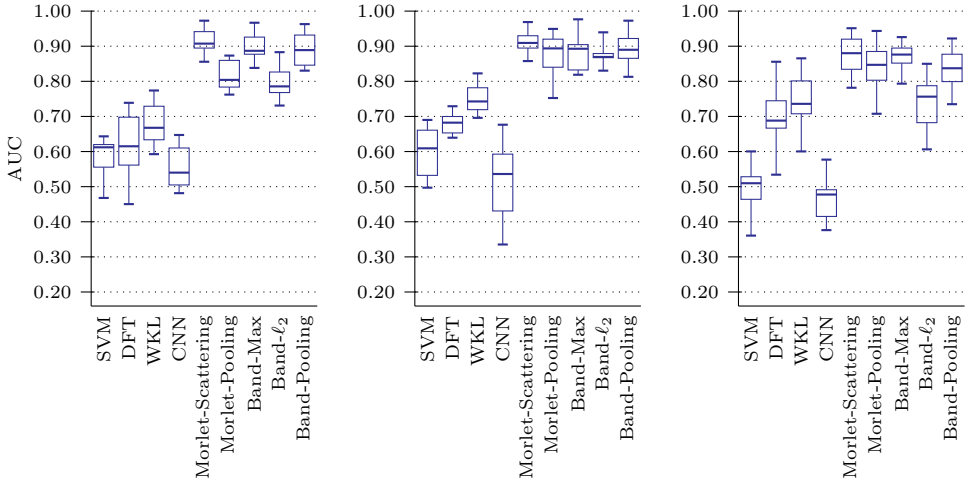
The approach presented here affords a new sight line to the problem of learning a discriminative FB, compared to a CNN. The main point of the approach is to work in a kernelized framework, which provides non-linear classifiers based on convex optimization (note that problem (2) could also be formulated differently, for instance through the SVM radius-margin bound [40] or through the kernel target alignment criterion [41], but this seems more natural this way and more convenient due to the non-convexity of the other formulations). On the contrary, MLP is based on a gradient descent of a non-convex objective. Consequences are noteworthy since, as it has been explained in the previous sections, the overall optimization scheme, we propose, handles a part of the intrinsic non-convexity of the problem through an inner randomization step, that makes it more stable than a randomly initialized gradient descents.

Besides, this approach proposes an answer to the main risk of CNNs, which is to overfit the training data, resulting in a poor ability to classify unknown signals. It can first occur with small training datasets. This phenomenon is often observed with CNNs (this is the case for instance in the toy dataset of Section 5.2), while our method is expected not to be subject to this drawback since it is based on SVM. Indeed, SVM tends to maximize the margin between both classes. Moreover, overfitting with CNNs appears because of the high complexity of the model (there are numerous parameters). On the contrary, as the filters we use are controlled by few parameters, our method is somehow regularized by the family of filters we chose and tends therefore to prevent overfitting.

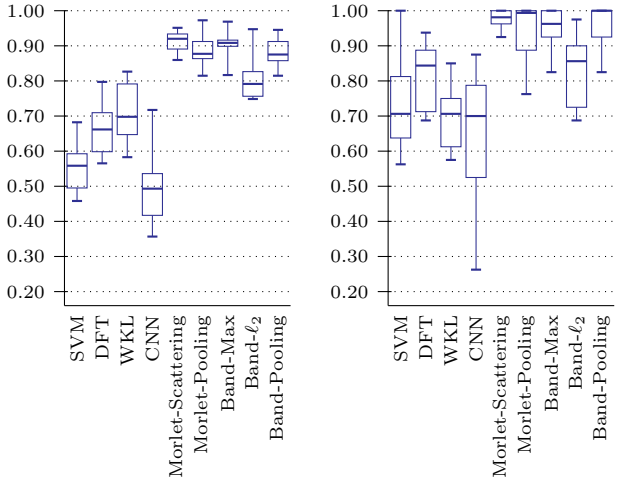
Finally, the proposed method offers several other conveniences. For instance, as gradients are computed with respect to filters weights, there is no need to consider smooth functions. Particularly, pooling functions like Max pooling can be handled without any concern. At the end, the proposed scheme is quite automated and does not need a deep experience to be tuned, unlike Neural Networks.

6.2. Infinite and Wavelet Kernel Learning

IKL [32] and WKL [6] are two distinct problems. While the first one is aimed at learning a multiple kernel (a convex combination) with potentially infinitely many kernels through Semi Infinite Linear Programming, the second one learns a combination of a huge number of kernels based on parts of wavelet decompositions, thanks to an active set method. Despite those differences, both problems share quite resembling algorithms, based on a column generation technique with a sparse MKL as inner problem. The major difference is the way to generate the column. While IKL tries to solve a non-convex problem, WKL samples some kernels until finding one that violates the optimality conditions.



(a) Channel 9. (b) Channel 12. (c) Channel 17.



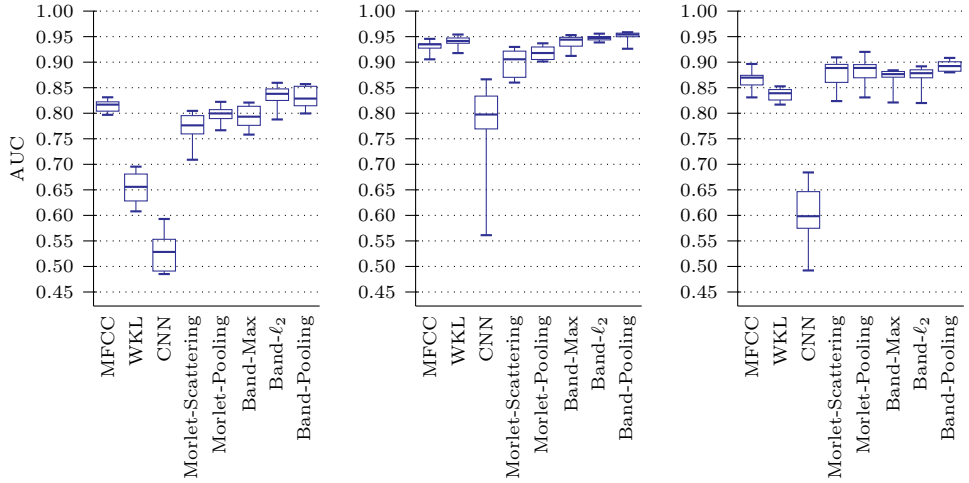
(d) Channel 29. (e) Channel 30.

Figure 7: BCI dataset: classification accuracy for several channels.

The work presented here is algorithmically inspired by both these contributions since we learn a product of potentially infinitely many kernels through an active set method. Yet, our approach is different from both IKL and WKL in the target: our work is firstly to learn a discriminative TF transform jointly with an SVM classifier. In addition, our approach has been driven in order that the resulting classification tool can be easily reduced to a two-stage line: first an FB and then an SVM classifier. This reduction is not possible with WKL, which can turn into a hindrance for interpreting the learned result. Note also that our algorithm turns out to be an extension of the non-linear MKL from [29] to an infinite amount of kernels. This problem can certainly not be addressed by IKL nor by WKL. Finally, even though there is no proof on convergence, we exhibit the strict decrease of the objective despite the non-convexity of the inner MKL problem when the kernel is Gaussian.

7. Conclusions

This paper has introduced a novel approach to learn a one-stage filter bank for signal classification. It took a fresh look at learning a discriminative time-frequency transform compared to the widespread convolutional neural networks. The method proposed to jointly learn a filter bank with an SVM classifier. The SVM kernels, we considered in this article, were the linear and the Gaussian ones. They were computed by filtering the signals thanks to the filter bank, and by pooling the result. We showed that, by choosing a specific family of filters, defined by few parameters, the optimization problem is actually a multiple kernel learning problem, where the number of kernels can be infinite. We thus provided an active constraint algorithm, that extended existing methods and that is able to handle such a problem by generating a sparse and finite non-necessarily convex combination of kernels. One advantage of our method, compared to neural networks, is



(a) Park *vs* quiet street. (b) Office *vs* supermarket. (c) Office *vs* quiet street.

Figure 8: Scenes dataset: classification accuracy.

that the number of filters has not to be chosen beforehand, it is automatically determined. Moreover, although it was not our main goal, we proved that the built framework also enables to learn a particular form of pooling function (and thus the translation-invariance of the kernels). Numerical experiments undertaken on a brain-computer interface dataset and on a scene classification problem showed that the proposed approach is competitive and provides a relevant alternative to existing methods. Additionally, learning the pooling function appeared to be an extra feature for the practitioner, who does not need to chose the kind of translation-invariance beforehand.

As a kernel learning method, the main limitation of our approach is the computational complexity. Even though finding a new coordinate in the active constraint algorithm is a highly parallelizable task, the algorithm is poorly scalable. The previous pitfall is related to the difficulty of selecting hyperparameters in kernel learning methods, which is still an open question. We are thus determined to devote some efforts in alleviating this pitfall.

Acknowledgments

We would like to thank the anonymous reviewers for contributing to the clarity of the presentation. This work was partially supported by the *Direction Générale de l'Armement* (French Ministry of Defense) and by the French ANR (09-EMER-001 and 12-BS03-003).

Appendix A. Optimality conditions

The principle of active set needs some optimality necessary conditions to have the current set evolve up to its final form. Despite the non-convexity of the MKL problem for the Gaussian kernel k, the Karush-Kuhn-Tucker (KKT) conditions can be used since they are necessary conditions for non-convex optimization programs. The forthcoming paragraphs are thus dedicated to derive a way to check the optimality of a given weighting vector $\boldsymbol{\mu}$, regarding problem (11).

Let us write a Lagrangian function L associated to (11), where λ is the Lagrange multiplier of the sparsity constraint $\mathbb{1}^T\boldsymbol{\mu} = 1$ and $\boldsymbol{\tau}$ is the dual vector of the non-negativeness constraint $\boldsymbol{\mu} \succcurlyeq 0$:

$$\begin{aligned}&\forall(\boldsymbol{\mu}, \lambda, \boldsymbol{\tau}) \in \mathbb{R}^{\mathcal{P}} \times \mathbb{R} \times \mathbb{R}^{\mathcal{P}},\\ &L(\boldsymbol{\mu}, \lambda, \boldsymbol{\tau}) = J\left(u_0, k_{[\boldsymbol{\mu}]}\right) + \lambda(\mathbb{1}^T\boldsymbol{\mu} - 1) - \boldsymbol{\tau}^T\boldsymbol{\mu}.\end{aligned}$$

At an equilibrium in the multiple kernel problem (potentially a local minimum or maximum), the KKT conditions hold:

$$\begin{cases} \mathbb{1}^T\mu = 1,\ \mu \succcurlyeq 0 \\ \boldsymbol{\tau} \succcurlyeq 0 \\ \tau_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}} = 0,\ \forall \boldsymbol{\theta} \in \mathcal{P} \\ \nabla_{\boldsymbol{\mu}} L(\boldsymbol{\mu}, \lambda, \boldsymbol{\tau}) = 0. \end{cases}$$

Both first conditions (primal and dual feasibility) are just reminders. The last one is however quite important and can be rewritten:

$$\nabla \tilde{J}(\boldsymbol{\mu}) + \lambda \mathbb{1} - \boldsymbol{\tau} = 0, \tag{16}$$

where

$$\tilde{J} \colon \boldsymbol{\mu} \in \mathbb{R}^{\mathcal{P}} \mapsto J\left(u_0, k_{[\boldsymbol{\mu}]}\right).$$

Combining the third KKT condition with (16) leads to:

$$\forall \boldsymbol{\theta} \in \mathcal{P},\ \begin{cases} \dfrac{\partial \tilde{J}}{\partial \mu_{\boldsymbol{\theta}}}(\boldsymbol{\mu}) = -\lambda,\ \text{if } \mu_{\boldsymbol{\theta}} > 0 \\ \dfrac{\partial \tilde{J}}{\partial \mu_{\boldsymbol{\theta}}}(\boldsymbol{\mu}) \geq -\lambda,\ \text{if } \mu_{\boldsymbol{\theta}} = 0. \end{cases} \tag{17}$$

Once the equilibrium is achieved, the Lagrange multiplier λ is given by:

$$\lambda \stackrel{\text{def}}{=} -\sum_{\boldsymbol{\theta} \in \mathcal{P},\, \mu_{\boldsymbol{\theta}} > 0} \mu_{\boldsymbol{\theta}} \frac{\partial \tilde{J}}{\partial \mu_{\boldsymbol{\theta}}}(\boldsymbol{\mu}).$$

It remains then to compute the partial derivatives of $\tilde{J}$. This is possible thanks to the theorem 4.1 from [42], which claims that since the SVM objective function is differentiable and has a unique minimizer (this is guaranteed if the kernel matrix is positive definite, which can be always true by adding a little ridge), the gradient of $\tilde{J}$ is given by the gradient of the SVM objective at the optimum. To compute it, let us write the SVM dual problem. Let $\boldsymbol{Y} \stackrel{\text{def}}{=} \text{diag}(\boldsymbol{y})$ and $\boldsymbol{K}_{[\boldsymbol{\mu}]_+}$ be the positive definite kernel matrix defined by:

$$\boldsymbol{K}_{[\boldsymbol{\mu}]_+} \stackrel{\text{def}}{=} (k_{[\boldsymbol{\mu}]}((\rho \circ u_0)(\boldsymbol{x_i}), (\rho \circ u_0)(\boldsymbol{x_j})))_{1 \leq i,j \leq N}.$$

In practice, instead of tackling problem (3), SVM solves the dual form (in which $\boldsymbol{\alpha}$ is the dual vector):

$$\begin{array}{l} \underset{\boldsymbol{\alpha} \in \mathbb{R}^N}{\texttt{maximize}} \ \mathbb{1}^T \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{Y} \boldsymbol{K}_{[\boldsymbol{\mu}]_+} \boldsymbol{Y} \boldsymbol{\alpha} \\ \texttt{s.t.} \quad 0 \preccurlyeq \boldsymbol{\alpha} \preccurlyeq C\mathbb{1}, \ \boldsymbol{y}^T \boldsymbol{\alpha} = 0. \end{array} \tag{18}$$

Thus from [42],

$$\forall \boldsymbol{\theta} \in \mathcal{P}, \ \frac{\partial \tilde{J}}{\partial \mu_{\boldsymbol{\theta}}}(\boldsymbol{\mu}) = -\frac{1}{2} \boldsymbol{\alpha^*}^T \boldsymbol{Y} \frac{\partial \boldsymbol{K}_{[\boldsymbol{\mu}]_+}}{\partial \mu_{\boldsymbol{\theta}}} \boldsymbol{Y} \boldsymbol{\alpha^*}, \tag{19}$$

where $\boldsymbol{\alpha^*}$ is the same as in (4) and (12). Let us compute the partial derivative of the kernel: as $k_{[\boldsymbol{\mu}]}$ is a product of spanning kernels,

$$k_{[\boldsymbol{\mu}]} \stackrel{\text{def}}{=} \prod_{\boldsymbol{\theta} \in \mathcal{P}} k_{\boldsymbol{\theta}}^{\mu_{\boldsymbol{\theta}}} = \mathrm{e}^{-\gamma \sum_{\boldsymbol{\theta} \in \mathcal{P}} \mu_{\boldsymbol{\theta}} d_{\boldsymbol{\theta}}},$$

where $d_{\boldsymbol{\theta}}$ is a similarity measure depending on $\boldsymbol{\theta}$, then:

$$\frac{\partial k_{[\boldsymbol{\mu}]}}{\partial \mu_{\boldsymbol{\theta}}} = -\gamma d_{\boldsymbol{\theta}} k_{[\boldsymbol{\mu}]}.$$

Now, let $\boldsymbol{D_\theta}$ be the similarity matrix:

$$\boldsymbol{D_\theta} \stackrel{\text{def}}{=} (d_{\boldsymbol{\theta}}(\boldsymbol{x_i}, \boldsymbol{x_j}))_{1 \leq i,j \leq N},$$

then the main equilibrium condition to be violated is:

$$\forall \boldsymbol{\theta} \in \mathcal{P}, \ -\frac{\partial \tilde{J}}{\partial \mu_{\boldsymbol{\theta}}}(\boldsymbol{\mu}) = -\frac{\gamma}{2} \boldsymbol{\alpha^*}^T \boldsymbol{Y} \left(\boldsymbol{D_\theta} \circ \boldsymbol{K}_{[\boldsymbol{\mu}]_+} \right) \boldsymbol{Y} \boldsymbol{\alpha^*} \leq \lambda.$$

References

[1] N. Saito, R. Coifman, Local discriminant bases and their applications, Journal of Mathematical Imaging and Vision 5 (1995) 337–358.

[2] D. Blei, J. McAuliffe, Supervised topic models, in: Advances in Neural Information Processing Systems (NIPS), MIT Press, 2008.

[3] R. Flamary, D. Tuia, B. Labbé, G. Camps-Valls, A. Rakotomamonjy, Large margin filtering, IEEE Transactions on Signal Processing 60 (2012) 648–659.

[4] E. Jones, P. Runkle, N. Dasgupta, L. Couchman, L. Carin, Genetic algorithm wavelet design for signal classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (2001) 890–895.

[5] D. J. Strauss, G. Steidl, W. Delb, Feature extraction by shape-adapted local discriminant bases, Signal Processing 83 (2003) 359–376.

[6] F. Yger, A. Rakotomamonjy, Wavelet kernel learning, Pattern Recognition 44 (2011) 2614–2629.

[7] M. Davy, A. Gretton, A. Doucet, P. J. W. Rayner, Optimized support vector machines for nonstationary signal classification, IEEE Signal Processing Letters 9 (2002) 442–445.

[8] P. Honeine, C. Richard, P. Flandrin, J.-B. Pothin, Optimal selection of time-frequency representations for signal classification: a kernel-target alignment approach, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2006.

[9] J. Mairal, F. Bach, J. Ponce, G. Sapiro, Zisserman, Supervised dictionary learning, in: Advances in Neural Information Processing Systems (NIPS), 2008.

[10] K. Huang, S. Aviyente, Sparse representation for signal classification, in: Advances in Neural Information Processing Systems (NIPS), 2006.

[11] F. Rodriguez, G. Sapiro, Sparse representation for image classification: Learning discriminative and reconstructive non-parametric dictionaries, Tech. rep., University of Minnesota (2008).

[12] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, IEEE Proceedings 86(11) (1998) 2278–2324.

[13] A. Biem, S. Katagiri, E. McDermott, G.-H. Juang, An application of discriminative feature extraction to filter-bank-based speech recognition, IEEE Transactions on Speech and Audio Processing 9 (2) (2001) 96–110.

[14] P. Vaidyanathan, Multirate systems and filter banks, Prentice Hall, 1993.

[15] H. Kha, H. Tuan, T. Nguyen, Efficient design of cosine-modulated filter banks via convex optimization, IEEE Transactions on Signal Processing 57 (2009) 966–976.

[16] J. Gauthier, L. Duval, J.-C. Pesquet, Optimization of synthesis oversampled complex filter banks, IEEE Transactions on Signal Processing 57 (2009) 3827–3843.

[17] L. D. Vignolo, H. L. Rufiner, D. H. Milone, J. Goddard, Evolutionary cepstral coefficients, Applied Soft Computing 11 (2011) 3419–3428.

[18] L. D. Vignolo, H. L. Rufiner, D. H. Milone, J. Goddard, Evolutionary splines for cepstral filterbank optimization in phoneme classification, EURASIP Journal on Advances in Signal Processing 2011 (2011) 1–14.

[19] H.-I. Suk, S.-W. Lee, A novel bayesian framework for discriminative feature extraction in brain-computer interfaces, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (2) (2013) 286–299.

[20] T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning: data mining, inference, and prediction, Springer, 2008.

[21] M. Sangnier, J. Gauthier, A. Rakotomamonjy, Filter bank kernel learning for nonstationary signal classification, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2013.

[22] G. Strang, T. Nguyen, Wavelets and filter banks, Wellesley-Cambridge Press, 1996.

[23] S. Mallat, A Wavelet Tour of Signal Processing: The Sparse Way, Elsevier Science, 2008.

[24] S. Mallat, Group invariant scattering, Communications in Pure and Applied Mathematics 65 (2012) 1331–1398.

[25] J. Andén, S. Mallat, Deep scattering spectrum, IEEE Transactions on Signal Processing.

[26] J. Shawe-Taylor, N. Cristianini, Kernel methods for pattern analysis, Cambridge University Press, 2004.

[27] A. Rakotomamonjy, F. Bach, S. Canu, Y. Grandvalet, SimpleMKL, Journal of Machine Learning Research 9 (2008) 2491–2521.

[28] V. Vapnik, Statistical learning theory, Wiley, 1998.

[29] M. Varma, B. Babu, More generality in efficient multiple kernel

learning, in: Proceedings of the 26th International Conference on Machine Learning, 2009.
[30] C.-C. Chang, C.-J. Lin, Libsvm: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (3) (2011) 27.
[31] M. Szafranski, Y. Grandvalet, A. Rakotomamonjy, Composite kernel learning, Machine Learning 79 (2010) 73–103.
[32] P. Gehler, S. Nowozin, Infinite kernel learning, in: Advances in Neural Information Processing Systems (NIPS), 2008.
[33] J. Nocedal, S. Wright, Numerical optimization, Springer, 2000.
[34] D. Luenberger, Linear and nonlinear programming, Addison-Wesley, 1984.
[35] Y. Boureau, F. Bach, L. Y., J. Ponce, Learning mid-level features for recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010.
[36] Y. Boureau, J. Ponce, L. Y., A theoretical analysis of feature pooling in visual recognition, in: Proceedings of the 27th International Conference on Machine Learning, 2010.
[37] Q. Barthélemy, M. Sangnier, A. Larue, J. Mars, Comparaison de descripteurs pour la classification de décompositions parcimonieuses invariantes par translation, in: XXIVème Colloque GRETSI, 2013.
[38] A. Rakotomamonjy, R. Flamary, F. Yger, Learning with infinitely many features, Machine Learning 91 (2013) 43–66.
[39] D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, M. Plumbley, Detection and classification of acoustic scenes and events, Tech. rep., Queen Mary University of London, an IEEE AASP Challenge (March 2013).
[40] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Machine Learning 46 (2002) 131–159.
[41] C. Cortes, M. Mohri, A. Rostamizadeh, Algorithms for learning kernels based on centered alignment, Journal of Machine Learning Research 13 (2012) 795–828.
[42] F. Bonnans, A. Shapiro, Optimization problems with perturbations, a guided tour, SIAM Journal on Scientific Computing 40 (1996) 228–264.