2014 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 21–24, 2014, REIMS, FRANCE

KERNEL LEARNING AS MINIMIZATION OF THE SINGLE VALIDATION ESTIMATE

*Maxime Sangnier, Jérôme Gauthier**

CEA, LIST
91191 Gif-Sur-Yvette, FRANCE
{first name}.{last name}@cea.fr

Alain Rakotomamonjy

University of Rouen
76800 Saint-Etienne du Rouvray, FRANCE
alain.rakoto@univ-rouen.fr

ABSTRACT

In order to prevent overfitting in traditional support vector kernel learning, we propose to learn a kernel (jointly with the cost parameter C) by minimizing the single validation estimate with a sequential linear filter algorithm. Additionally, we introduce a simple heuristic in order to improve risk estimation, which randomly swaps several points between the validation and the training sets. Contrarily to previous works, which use several validation sets to improve risk estimation, our strategy does not increase the number of optimization variables. This is easily done thanks to Karasuyama and Takeuchi's multiple incremental decremental support vector learning algorithm. A synthetic signal classification problem underlines the effectiveness of our method. The main parameters of the learned kernel are the finite impulse responses of a filter bank.

***Index Terms*—** Support vector machine, kernel learning, complementary constraint, bilevel optimization, filter bank.

1. INTRODUCTION

In statistical learning, kernel methods are popular and successful tools [1]. In particular, support vector learning has been applied with great success to classification tasks. Such performance led researchers to build feature selection algorithms upon support vector machine-based criteria [2]. In parallel to feature selection, a wide interest has been devoted to kernel learning (for the same purpose), *i.e.* the way to jointly learn a classifier as a linear functional, and the space in which the latter lies [3, 4]. While kernel methods are usually based on convex optimization (thus theoretically and computationally tractable), kernel learning is generally embodied by non-convex and time-consuming optimization problems. The last but not least drawback of this kind of method is to overfit the training data when there are many kernel parameters.

Kernel learning has been widely studied in the 2000's. The contributions can be clustered in different manners. For instance, we may distinguish the direct, target and generalized approaches. The first ones learn a kernel by minimizing a criterion based on the structural risk of misclassification [5, 6]. They are direct extensions of support vector principles to kernel learning and have been proved to be quite efficient for special applications [6]. The target methods learn a kernel by fitting it with an ideal one [7]. Thus, they are composed of two stages (first fitting the kernel and then learning the classifier). This is advantageous compared to direct approaches, which generally solve a higher dimensional problem [5] or solve many quadratic problems in an inner loop [6]. Eventually, generalized approaches focus on minimizing a generalization bound of the misclassification error [8, 9, 10]. These methods are aimed at preventing overfitting while learning a support vector kernel. While some of the direct and target methods have been proved to be based on convex optimization [6, 7], all the generalized approaches solve locally a non-convex problem.

All the previously mentioned approaches are very specific to kernel methods. They take advantage from the particular structure of the support vector problem, which is based on a positive-definite similarity matrix and which sparsely selects discriminative vectors. On the contrary, a more general method has been introduced for the problem of model selection for support vector learning in [11, 12]. The principle of these latter works is to tackle the problem of multiple validation estimate minimization as a bilevel program. This criterion, which is the most common instance of generalization bound, is shown in [11, 12] to be simple to cast as a minimization problem and to be efficiently solved by off-the-shelf solvers. In comparison, tackling the same problem with a grid search (*i.e.* performing a cross-validation) would be prohibitive.

Outline In this study, we propose to deal with the single validation estimate minimization with an ℓ_1-regularized support vector machine. Our work is aimed at providing a new kernel learning method, which prevents overfitting when there are many kernel parameters compared to the size of the training dataset. Moreover, in order to improve the estimation without increasing the number of variables (which occurs when several validation sets are used, like in [11, 12]), we propose a simple heuristic. This one consists in swapping several points between the validation and the training sets during the descent optimization. Additionally, these swaps are made efficient by using incremental techniques of support vector classification [13]. This makes our approach a little more specific to kernel learning than [11, 12]. Eventually, our approach demonstrates promising results on a signal classification problem based on a synthetic toy dataset. In this numerical experiment, the kernel that we design is built upon a filter bank for which finite impulse responses are not known beforehand and have thus to be learned.

Relation to prior work The approach presented in this paper is related to several previous works. The first one is [8], in which the authors minimize the single validation estimate for an ℓ_2-regularized support vector machine. On the contrary, we address the problem with an ℓ_1-regularized classifier. [9] extends the method from [8] to multiple validation sets, ℓ_1-regularized support vector machines and focuses on building an efficient (*i.e.* quick) gradient-based algorithm. This approach is clearly among the state of the art in gen-

*Thanks to the *Direction Générale de l'Armement*, French Ministry of Defense, for funding.

978-1-4799-3694-6/14/$31.00

eralized kernel learning and we do not try to challenge it. On the contrary, we aim to remind that other ways to minimize the multiple validation estimate exist. For small datasets, for which overfitting occurs, our method is not long to train, even if we use basic techniques to compute the gradient of the kernel (which is needed to linearize the constraints of our learning problem).

Due to techniques from bilevel optimization, our approach has a similar flavor to that work published in [11, 12]. The particularity of our approach lies in not increasing the number of optimization variables by using several validation sets (and then several classifiers). Furthermore, we tried to demonstrate that techniques from bilevel optimization are not just easy to use and workable because of the existence of off-the-shelf optimization software (as the ones used in [11, 12]), but are also accessible to implement and can be made efficient by using tricks, like the proposed heuristic.

2. KERNELIZED FRAMEWORK FOR CLASSIFICATION

2.1. Support vector machine

Let $\Big((\boldsymbol{x_i}, y_i)\Big)_{i\in\mathcal{S}}$ be a dataset, where each data $\boldsymbol{x_i}$ comes from a compact set and each label y_i is either $+1$ or -1 (we consider a biclass problem). Let also $\{\mathcal{V}, \mathcal{T}\}$ be a partition of the set of indexes $\mathcal{S}$. In the whole paper, $\mathcal{T}$ describes the training dataset and $\mathcal{V}$ the validation one.

Let $\mathcal{H}_{\boldsymbol{\theta}}$ be a reproducing kernel Hilbert space (generated by a kernel $k_{\boldsymbol{\theta}}$) parametrized by a d-dimensional vector $\boldsymbol{\theta}$ and C a cost parameter ($C \geq 0$). The structural risk of misclassification [14] is defined by:

$$J_{C,\boldsymbol{\theta}}(f, b) \stackrel{\text{def}}{=} \frac{1}{2}\|f\|^2_{\mathcal{H}_{\boldsymbol{\theta}}} + C\sum_{i\in\mathcal{T}} \Lambda(f(\boldsymbol{x_i}) + b, y_i),$$

where f is a functional from $\mathcal{H}_{\boldsymbol{\theta}}$ and b is a bias from $\mathbb{R}$. Λ is a cost function, which reflects the discrepancy between the predicted value $f(\boldsymbol{x_i}) + b$ and the true label y_i.

Given some parameters C and $\boldsymbol{\theta}$, a support vector machine is an algorithm that learns the prediction function $\boldsymbol{x} \mapsto f(\boldsymbol{x}) + b$ by solving the variational problem:

$$\underset{f\in\mathcal{H}_{\boldsymbol{\theta}}, b\in\mathbb{R}}{\texttt{minimize}}\ J_{C,\boldsymbol{\theta}}(f, b).$$

Commonly, we use the non-differentiable hinge loss function

$$\Lambda\colon (a, b) \in \mathbb{R}^2 \to \texttt{max}\,(0, 1 - ab) \in \mathbb{R}_+,$$

which promotes a sparse selection of support vectors. With this loss function, the learning problem becomes:

$$\begin{array}{ll} \underset{f\in\mathcal{H}_{\boldsymbol{\theta}}, b\in\mathbb{R}, \boldsymbol{\xi}\in\mathbb{R}^{|\mathcal{T}|}}{\texttt{minimize}} & \frac{1}{2}\|f\|^2_{\mathcal{H}_{\boldsymbol{\theta}}} + C\mathbb{1}^T\boldsymbol{\xi} \\ \texttt{s.t.} & \forall i \in \mathcal{T}, 1 - \xi_i - y_i(f(\boldsymbol{x_i}) + b) \leq 0 \\ & 0 \preccurlyeq \boldsymbol{\xi}, \end{array} \tag{1}$$

where $\mathbb{1}$ stands for the all-one vector. Let $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ from $\mathbb{R}^{|\mathcal{T}|}$ be respectively the Lagrangian vectors for the first and the second constraint of (1). According to the Karush-Kuhn-Tucker conditions, $(f, b, \boldsymbol{\xi})$ is a solution of (1) iff for some $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$

$$\left\{\begin{array}{l} \forall i \in \mathcal{T}, 1 - \xi_i - y_i(f(\boldsymbol{x_i}) + b) \leq 0 \\ 0 \preccurlyeq \boldsymbol{\xi} \\ 0 \preccurlyeq \boldsymbol{\alpha} \\ 0 \preccurlyeq \boldsymbol{\lambda} \\ \forall i \in \mathcal{T}, \lambda_i\xi_i = 0 \\ \forall i \in \mathcal{T}, (1 - \xi_i - y_i(f(\boldsymbol{x_i}) + b))\alpha_i = 0 \\ f = \sum_{i\in\mathcal{T}} \alpha_i y_i k_{\boldsymbol{\theta}}(\boldsymbol{x_i}, \cdot) \\ \boldsymbol{\alpha}^T\boldsymbol{y_\mathcal{T}} = 0 \\ \boldsymbol{\lambda} = C\mathbb{1} - \boldsymbol{\alpha}, \end{array}\right. \tag{2}$$

where $\boldsymbol{y_\mathcal{T}}$ is the vector of training labels.

2.2. Kernel learning

Problem (1) is easily solvable if the trade-off parameter C and the kernel parameter $\boldsymbol{\theta}$ are known beforehand [15, 16]. Yet it is never the case. Hence, when $\boldsymbol{\theta}$ has at most two components, these parameters are usually chosen by cross-validation. A more general approach is to learn the kernel parameter $\boldsymbol{\theta}$ by solving a variational problem [17, 5] and to perform a cross-validation in order to find the trade-off parameter C. So far, the easiest way to do so is to tackle the problem

$$\begin{array}{ll} \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\texttt{minimize}} & \left(\min_{f\in\mathcal{H}_{\boldsymbol{\theta}}, b\in\mathbb{R}} J_{C,\boldsymbol{\theta}}(f, b)\right) + \rho(\boldsymbol{\theta}) \\ \texttt{s.t.} & \kappa(\boldsymbol{\theta}) \leq 0 \end{array}$$

by a first-order approach (ρ and κ being respectively a regularization and a constraint function from $\mathbb{R}^d$ to $\mathbb{R}$). This turns out to be one of the most efficient way to learn a kernel when the latter lies in the smallest convex set containing some predefined kernels [6].

2.3. Filter bank kernel

As we are interested in signal classification, we consider a processing line built upon three consecutive stages: i) filtering by a filter bank with all the decimation factors set to 1; ii) pooling with an energy-based function that partitions the filtered signal into a set of non-overlapping segments (of length w) and outputs the energy of each such sub-signal; iii) mapping with a Gaussian kernel of parameter γ ($\gamma \geq 0$). Eventually, the kernel that we consider is

$$k\colon (\boldsymbol{x_1}, \boldsymbol{x_2}) \mapsto \exp\left(-\gamma\|P(T(\boldsymbol{x_1})) - P(T(\boldsymbol{x_2}))\|_2^2\right),$$

where $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are signals and the time-frequency transformation performed by the filter bank is given by the function T ($*$ being the convolution operation) based on m finite impulse response filters $\boldsymbol{h_k}$:

$$T(\boldsymbol{x}) \stackrel{\text{def}}{=} [\ldots, \boldsymbol{h_k} * \boldsymbol{x}, \ldots]^T_{1\leq k\leq m},$$

and the general pooling operator is P:

$$P(T(\boldsymbol{x})) \stackrel{\text{def}}{=} \Big[\ldots, \tilde{P}(\boldsymbol{h_k} * \boldsymbol{x}), \ldots\Big]^T_{1\leq k\leq m},$$

with the core function $\tilde{P}$, which returns the local energy of a filtered signal $\tilde{\boldsymbol{x}} \stackrel{\text{def}}{=} \boldsymbol{h_k} * \boldsymbol{x}$, defined by:

$$\tilde{P}(\tilde{\boldsymbol{x}}) \stackrel{\text{def}}{=} \left[\ldots, \sum_{k=1}^{w} |\tilde{x}_{iw+k}|^2, \ldots\right]^T_{0\leq i\leq \left\lfloor \frac{|\tilde{\boldsymbol{x}}|}{w}\right\rfloor}.$$

With these definitions, the kernel parameter $\boldsymbol{\theta}$ is $\boldsymbol{\theta} \stackrel{\text{def}}{=} (\gamma, \boldsymbol{h_1}^T, \ldots, \boldsymbol{h_m}^T)^T$. Following [6, 3], a first approach

to learn such a kernel would be to find a local minimum to the problem

$$\begin{array}{ll} \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\texttt{minimize}} & \left(\min_{f\in\mathcal{H}_{\boldsymbol{\theta}},b\in\mathbb{R}} J_{C,\boldsymbol{\theta}}(f,b)\right) \\ \texttt{s.t.} & 0\leq\gamma \\ & \|\boldsymbol{h_1}\|_2^2+\cdots+\|\boldsymbol{h_m}\|_2^2\leq 1. \end{array} \tag{3}$$

The first constraint ensures the positive-definiteness of the kernel k and the second excludes a degeneracy. We could have used other kind of kernels, and especially other pooling functions $\tilde{P}$. Nevertheless, in order to be suitable to our method, the kernel must be differentiable with respect to all its parameters.

3. KERNEL LEARNING THROUGH BILEVEL OPTIMIZATION

3.1. Problem formulation

Ideally, the best way to prevent overfitting is to minimize the true risk of the classifier. Since this risk is unavailable, we choose to minimize an estimate of it computed as the empirical error on a validation set $\mathcal{V}$ (this is the single validation estimate):

$$\begin{array}{ll} \underset{C\in\mathbb{R}_+,\boldsymbol{\theta}\in\mathbb{R}^d}{\texttt{minimize}} & \sum_{i\in\mathcal{V}}\Lambda(f_{\boldsymbol{\theta}}(\boldsymbol{x_i})+b_{\boldsymbol{\theta}},y_i) \\ \texttt{s.t.} & (f_{\boldsymbol{\theta}},b_{\boldsymbol{\theta}})\in\underset{f\in\mathcal{H}_{\boldsymbol{\theta}},b\in\mathbb{R}}{\operatorname{argmin}} J_{C,\boldsymbol{\theta}}(f,b). \end{array} \tag{4}$$

In order to solve problem (4), we circumvent the variational constraint by substituting the lower-level problem (which is a standard support vector learning) by its Karush-Kuhn-Tucker optimality conditions (2):

$$\begin{array}{ll} \underset{C,\boldsymbol{\theta},b,\boldsymbol{\xi},\boldsymbol{\alpha}}{\texttt{minimize}} & \sum_{i\in\mathcal{V}}\Lambda(f(\boldsymbol{x_i})+b,y_i) \\ \texttt{s.t.} & 0\preccurlyeq\boldsymbol{\xi} \\ & 0\preccurlyeq\boldsymbol{\alpha}\preccurlyeq C\mathbb{1} \\ & \boldsymbol{\alpha}^T\boldsymbol{y_{\mathcal{T}}}=0 \\ & f=\sum_{i\in\mathcal{T}}\alpha_i y_i k_{\boldsymbol{\theta}}(\boldsymbol{x_i},\cdot) \\ \forall i\in\mathcal{T} & \left\{\begin{array}{l} 1-\xi_i-y_i(f(\boldsymbol{x_i})+b)\leq 0 \\ (C-\alpha_i)\xi_i=0 \\ (1-\xi_i-y_i(f(\boldsymbol{x_i})+b))\alpha_i=0. \end{array}\right. \end{array} \tag{5}$$

Problem (5) is not only nonlinearly constrained, but also nonconvex. This is due to the last two constraints, also known as complementarity constraints, which give rise to both theoretical and computational anomalies [11]. Several proposals can deal with those constraints (like branch and bound or penalty methods [18]) and particularly relaxation:

$$\forall i\in\mathcal{T}\left\{\begin{array}{l} (C-\alpha_i)\xi_i\leq\epsilon \\ (y_i(f(\boldsymbol{x_i})+b)+\xi_i-1)\alpha_i\leq\epsilon, \end{array}\right.$$

where ϵ is a positive tolerance threshold. Note that we do not need absolute values since each part of the products is non-negative. At this point, the only issue to alleviate is the non-differentiability of the hinge loss Λ in the objective function, which is easily done by introducing a slack vector $\boldsymbol{\mu}$. Before writing down the final optimization problem, let us introduce the matrices of training labels $\boldsymbol{Y}_+^{\mathcal{T}}\overset{\text{def}}{=}\operatorname{Diag}((y_i)_{i\in\mathcal{T}})$ and of validation labels $\boldsymbol{Y}_+^{\mathcal{V}}\overset{\text{def}}{=}\operatorname{Diag}((y_i)_{i\in\mathcal{V}})$, as well as the positive-definite kernel matrices for the training data $\boldsymbol{K}_{\boldsymbol{\theta}}^{\mathcal{T}}{}_+\overset{\text{def}}{=}(k_{\boldsymbol{\theta}}(\boldsymbol{x_i},\boldsymbol{x_j}))_{i,j\in\mathcal{T}}$ and for the validation data $\boldsymbol{K}_{\boldsymbol{\theta}}^{\mathcal{V}}\overset{\text{def}}{=}(k_{\boldsymbol{\theta}}(\boldsymbol{x_i},\boldsymbol{x_j}))_{i\in\mathcal{V},j\in\mathcal{T}}$. With these definitions, the problem to solve is:

$$\begin{array}{ll} \underset{C,\boldsymbol{\theta},\boldsymbol{\mu},b,\boldsymbol{\xi},\boldsymbol{\alpha}}{\texttt{minimize}} & \mathbb{1}^T\boldsymbol{\mu} \\ \texttt{s.t.} & 0\leq C \\ & \mathbb{1}-\boldsymbol{\mu}-\boldsymbol{Y}_+^{\mathcal{V}}(\boldsymbol{K}_{\boldsymbol{\theta}}^{\mathcal{V}}\boldsymbol{Y}_+^{\mathcal{T}}\boldsymbol{\alpha}+b\mathbb{1})\preccurlyeq 0 \\ & 0\preccurlyeq\boldsymbol{\mu} \\ & \mathbb{1}-\boldsymbol{\xi}-\boldsymbol{Y}_+^{\mathcal{T}}(\boldsymbol{K}_{\boldsymbol{\theta}}^{\mathcal{T}}{}_+\boldsymbol{Y}_+^{\mathcal{T}}\boldsymbol{\alpha}+b\mathbb{1})\preccurlyeq 0 \\ & 0\preccurlyeq\boldsymbol{\xi} \\ & 0\preccurlyeq\boldsymbol{\alpha}\preccurlyeq C\mathbb{1} \\ & \boldsymbol{\alpha}^T\boldsymbol{y_{\mathcal{T}}}=0 \\ \forall i\in\mathcal{T} & \left\{\begin{array}{l} (C-\alpha_i)\xi_i\leq\epsilon \\ (\boldsymbol{Y}_+^{\mathcal{T}}(\boldsymbol{K}_{\boldsymbol{\theta}}^{\mathcal{T}}{}_+\boldsymbol{Y}_+^{\mathcal{T}}\boldsymbol{\alpha}+b\mathbb{1})+\boldsymbol{\xi}-\mathbb{1})_i\alpha_i\leq\epsilon. \end{array}\right. \end{array} \tag{6}$$

3.2. Improving risk estimation

Even though the single validation estimate is unbiased and if its variance decreases with the size of the validation set [8], the size of the latter is limited in practice and even small in our experiments. Indeed overfitting usually occurs for small training datasets. In order to overcome that drawback, the cross-validation method minimizes the average empirical error on different non-overlapping validation sets, using a grid search strategy. This heuristic improves risk estimation but forces to learn many more classifiers. In [11, 12], the authors solve a variant of (6) with three validation sets, which seems to be an acceptable trade-off between estimation and computational time.

In this work, we introduce a novel heuristic to improve risk estimation without increasing the number of optimization variables (*i.e.* the number of learned classifiers). First, note that problem (6) will be solved by an iterative algorithm. Then, our heuristic consists in swapping several points between the validation and the training sets during the descent. In this way, the single learned classifier is evaluated on several validation sets and may not overfit. In practice, at a given iteration, we apply the swap procedure given by algorithm 1. This operation is repeated at a constant interval of iterations.

Data: a partition $\{\mathcal{V},\mathcal{T}\}$ of $\mathcal{S}$, a number n of points to swap.
Result: a new partition $\{\mathcal{V}^*,\mathcal{T}^*\}$ of $\mathcal{S}$.

$\mathcal{V}^\dagger\leftarrow$ random sample of size n from $\mathcal{V}$;
$\mathcal{T}^\dagger\leftarrow$ random sample of size n from $\mathcal{T}$;
$\mathcal{V}^*\leftarrow(\mathcal{V}\backslash\mathcal{V}^\dagger)\cup\mathcal{T}^\dagger$;
$\mathcal{T}^*\leftarrow(\mathcal{T}\backslash\mathcal{T}^\dagger)\cup\mathcal{V}^\dagger$;

Algorithm 1: Swap procedure.

To solve problem (6), we will decrease $\mathbb{1}^T\boldsymbol{\mu}$ while keeping satisfied the Karush-Kuhn-Tucker optimality conditions for the classifier (which are given in the constraints). Reciprocally, we would like to keep these constraints satisfied when swapping points between the evaluation and the training sets. Note that, when swapping points, the model parameters C and $\boldsymbol{\theta}$ are frozen. Besides, when knowing the new partition $\{\mathcal{V}^*,\mathcal{T}^*\}$, the vectors $\boldsymbol{\mu}$ and $\boldsymbol{\xi}$ are given by:

$$\begin{aligned} \boldsymbol{\mu} &= \max\left(0,\mathbb{1}-\boldsymbol{Y}_+^{\mathcal{V}^*}(\boldsymbol{K}_{\boldsymbol{\theta}}^{\mathcal{V}^*}\boldsymbol{Y}_+^{\mathcal{T}^*}\boldsymbol{\alpha}+b\mathbb{1})\right) \\ \boldsymbol{\xi} &= \max\left(0,\mathbb{1}-\boldsymbol{Y}_+^{\mathcal{T}^*}(\boldsymbol{K}_{\boldsymbol{\theta}}^{\mathcal{T}^*}{}_+\boldsymbol{Y}_+^{\mathcal{T}^*}\boldsymbol{\alpha}+b\mathbb{1})\right). \end{aligned}$$

The main difficulty is for the training variables $\boldsymbol{\alpha}$ and b. As suggested in algorithm 1, the swap can be split into two simultaneous steps: removing some points $\mathcal{T}^\dagger$ from $\mathcal{T}$, and adding some

other points $\mathcal{V}^\dagger$. This can be done thanks to Karasuyama and Takeuchi's multiple incremental and decremental learning algorithm [13], which provides us with the variables $\boldsymbol{\alpha}$ and b at the new equilibrium. Let us quickly review the core of the algorithm.

A dual of learning problem (1) is the minimax formulation

$$\underset{b\in\mathbb{R}}{\texttt{maximize}} \quad \underset{\boldsymbol{\alpha}\in\mathbb{R}^{|\mathcal{T}|},\, 0\preccurlyeq\boldsymbol{\alpha}\preccurlyeq C\mathbb{1}}{\texttt{minimize}} \; G(\boldsymbol{\alpha}, b), \tag{7}$$

with

$$G(\boldsymbol{\alpha}, b) \stackrel{\text{def}}{=} \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{Y}_+^{\mathcal{T}} \boldsymbol{K}_{\boldsymbol{\theta}}^{\mathcal{T}}{}_+ \boldsymbol{Y}_+^{\mathcal{T}} \boldsymbol{\alpha} - \mathbb{1}^T\boldsymbol{\alpha} + b\boldsymbol{y}_{\mathcal{T}}{}^T\boldsymbol{\alpha}.$$

Expression (7) is a saddle-point problem for which optimality is given by the Karush-Kuhn-Tucker conditions:

$$\begin{aligned}
&\forall i \in \mathcal{T}, \\
&\quad g_i \stackrel{\text{def}}{=} \frac{\partial G}{\partial \alpha_i}(\boldsymbol{\alpha}, b) \\
&\quad\quad = y_i\left(f(\boldsymbol{x_i}) + b\right) - 1 \begin{cases} \geq 0 & \text{if } \alpha_i = 0 \\ = 0 & \text{if } 0 < \alpha_i < C \\ \leq 0 & \text{if } \alpha_i = C \end{cases} \\
&\frac{\partial G}{\partial b}(\boldsymbol{\alpha}, b) = \quad \boldsymbol{y}_{\mathcal{T}}{}^T\boldsymbol{\alpha} = 0.
\end{aligned} \tag{8}$$

First, consider that the points we want to add to the training set are zero-weighted: $\boldsymbol{\alpha}_{\mathcal{V}^\dagger} = 0$. Secondly, let us keep in $\mathcal{V}^\dagger$ only the indexes of points that violate the optimality conditions (8) (the others can move to $\mathcal{T}$ without disrupting the equilibrium) and in $\mathcal{T}^\dagger$ only the indexes of points that support the model (*i.e.* $\alpha_i > 0$). Let us now partition the training dataset $\mathcal{T}$ into

$$\begin{aligned}
\mathcal{I} &\stackrel{\text{def}}{=} \left\{i \in \mathcal{T}\backslash\mathcal{T}^\dagger,\; y_i\left(f(\boldsymbol{x_i}) + b\right) < 1\right\}, \alpha_i = C \\
\mathcal{M} &\stackrel{\text{def}}{=} \left\{i \in \mathcal{T}\backslash\mathcal{T}^\dagger,\; y_i\left(f(\boldsymbol{x_i}) + b\right) = 1\right\}, 0 \leq \alpha_i \leq C \\
\mathcal{O} &\stackrel{\text{def}}{=} \left\{i \in \mathcal{T}\backslash\mathcal{T}^\dagger,\; y_i\left(f(\boldsymbol{x_i}) + b\right) > 1\right\}, \alpha_i = 0.
\end{aligned}$$

$\mathcal{I}$, $\mathcal{M}$ and $\mathcal{O}$ respectively describe the points inside, on the border and outside the margin (apart from the ones we want to remove). Note that, if $\boldsymbol{Q}_+ \stackrel{\text{def}}{=} \boldsymbol{Y}_+^{\mathcal{T}} \boldsymbol{K}_{\boldsymbol{\theta}}^{\mathcal{T}}{}_+ \boldsymbol{Y}_+^{\mathcal{T}}$, then g_i is expressed by

$$g_i = \sum_{j\in\mathcal{I}\cup\mathcal{M}\cup\mathcal{T}^\dagger} Q_{ij}\alpha_j + y_i b - 1.$$

Let Δ denote the amount of change of each variable (we assume it to be very small for now, such that points do not cross the frontiers of $\mathcal{I}$, $\mathcal{M}$ and $\mathcal{O}$). Then, to keep the optimality conditions given in (8), any change must satisfy the equalities

$$\begin{aligned}
&\forall i \in \mathcal{M},\; \Delta g_i = \textstyle\sum_{j\in\mathcal{M}} Q_{ij}\Delta\alpha_j + \sum_{j\in\mathcal{T}^\dagger} Q_{ij}\Delta\alpha_j \\
&\qquad\qquad + \textstyle\sum_{j\in\mathcal{V}^\dagger} Q_{ij}\Delta\alpha_j + y_i\Delta b = 0 \\
&\boldsymbol{y}_{\mathcal{M}}{}^T\boldsymbol{\Delta\alpha}_{\mathcal{M}} + \boldsymbol{y}_{\mathcal{T}^\dagger}{}^T\boldsymbol{\Delta\alpha}_{\mathcal{T}^\dagger} + \boldsymbol{y}_{\mathcal{V}^\dagger}{}^T\boldsymbol{\Delta\alpha}_{\mathcal{V}^\dagger} = 0
\end{aligned} \tag{9}$$

and the inequalities

$$\begin{aligned}
&\forall i \in \mathcal{I},\; g_i + \Delta g_i < 0 \\
&\forall i \in \mathcal{O},\; g_i + \Delta g_i > 0 \\
&\forall i \in \mathcal{M}\cup\mathcal{T}^\dagger\cup\mathcal{V}^\dagger,\; 0 \leq \alpha_i + \Delta\alpha_i \leq C.
\end{aligned} \tag{10}$$

Moreover, the points to add must still violate the equilibrium (otherwise they should have been added to $\mathcal{I}$, $\mathcal{M}$ or $\mathcal{O}$):

$$\forall i \in \mathcal{V}^\dagger,\; g_i + \Delta g_i < 0. \tag{11}$$

Now, if we choose the following directions of change

$$\boldsymbol{\Delta\alpha}_{\mathcal{V}^\dagger} = \eta\left(C\mathbb{1} - \boldsymbol{\alpha}_{\mathcal{V}^\dagger}\right), \quad \boldsymbol{\Delta\alpha}_{\mathcal{T}^\dagger} = -\eta\boldsymbol{\alpha}_{\mathcal{T}^\dagger},$$

where η is a non-negative step length, then any change $\boldsymbol{\Delta\alpha}_{\mathcal{V}^\dagger}$ and $\boldsymbol{\Delta\alpha}_{\mathcal{T}^\dagger}$ impacts b and $\boldsymbol{\alpha}_{\mathcal{M}}$ according to

$$\begin{bmatrix} \Delta b \\ \boldsymbol{\Delta\alpha}_{\mathcal{M}} \end{bmatrix} = \eta M \begin{bmatrix} C\mathbb{1} - \boldsymbol{\alpha}_{\mathcal{V}^\dagger} \\ -\boldsymbol{\alpha}_{\mathcal{T}^\dagger} \end{bmatrix},$$

where M is given by the two equalities in (9). The step length η is computed as the biggest one satisfying the inequalities from (10) and (11). This maximum step length matches with a point that moves across $\mathcal{I}$, $\mathcal{M}$, $\mathcal{O}$ or $\mathcal{V}^\dagger$. Thus, the algorithm in [13] updates these sets and then finds a new step length η. These two operations are repeated until $\mathcal{V}^\dagger$ is empty and $\boldsymbol{\alpha}_{\mathcal{T}^\dagger} = 0$.

3.3. Filter algorithm

In order to solve problem (6), we implemented a sequential linear programming filter method (see algorithm 2) [19]. Filter approaches were introduced in response to the difficulty of choosing suitable regularization parameters in penalty methods [20]. Their principle is close to Pareto multi-objective optimization, trying to minimize both the objective function and a measure of constraint violation, but with a priority on the latter, in that it is essential to find a solution that corresponds to a feasible point of (6).

In order to understand filter methods, we need to explain the concept of domination. Let two points generated during the process of solving problem (6) and represented by two pairs (h, obj), where h is a measure of constraint violation and obj is the current objective value. We say that (h_1, obj_1) dominates (h_2, obj_2) iff $(h_1, obj_1) \preccurlyeq (h_2, obj_2)$, *i.e.* there is no reason to prefer (h_2, obj_2) compared to (h_1, obj_1). A filter is a list of pairs (h_k, obj_k) such that no pair dominates any other.

On every iteration, we perform a new step by solving a linearized version of (6) within a trust region of size ρ (this is the sequential linear part of the algorithm). This is made possible by introducing the additional constraints (12), where Δ still denotes the amount of change of each variable:

$$\begin{aligned}
&\|\Delta C\|_\infty \leq \rho,\; \|\boldsymbol{\Delta\theta}\|_\infty \leq \rho,\; \|\boldsymbol{\Delta\mu}\|_\infty \leq \rho, \\
&\|\Delta b\|_\infty \leq \rho,\; \|\boldsymbol{\Delta\xi}\|_\infty \leq \rho,\; \|\boldsymbol{\Delta\alpha}\|_\infty \leq \rho.
\end{aligned} \tag{12}$$

If the resulting pair (h, obj) is acceptable to the filter (*i.e.* it is not dominated by any of the entries in the filter), then we linearly update the optimization variables as well as the filter (adding the new pair (h, obj) and removing any entries that are dominated by it) and the trust region radius ρ (increasing it by two). On the other hand, if the pair is not acceptable, then we reject it, reduce the trust region radius ρ by half and solve the linear program again.

4. NUMERICAL EXPERIMENTS

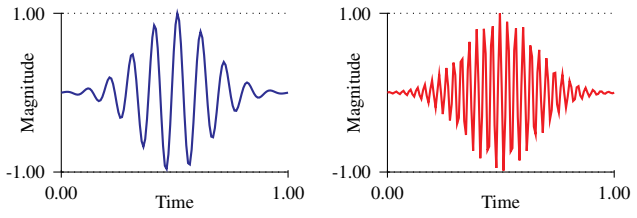
The algorithm proposed in this paper has been evaluated with a toy dataset. This one is made up of two classes based on different patterns (figure 1). The first pattern is a sine curve with normalized frequency that varies around 0.078 (10 Hz for a 128 Hz sampling rate). The second one is another sine curve with higher normalized frequency, which varies around 0.234 (30 Hz). 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 and the latter is tapered with a Gaussian window (which also follows the random time-shift). The colored noise comes from a Gaussian white noise, which is randomly filtered by either $[1, -2, 1]$, $[0, 1, -1]$ or $[1, 0, 1]$. This noise is stationary at the

Data: a training dataset $(\boldsymbol{x_i}, y_i)_{i\in\mathcal{S}}$, a number n of points to swap and an interval N of iterations.
Result: the model parameters C and $\boldsymbol{\theta}$.

Randomly partition $\mathcal{S}$ in $\{\mathcal{V}, \mathcal{T}\}$;
Initialize variables C and $\boldsymbol{\theta}$;
Initialize other variables with a single batch support vector learning;
Initialize the trust region radius ρ;
while *improvement is possible* **do**
    $obj \leftarrow$ solve a linearized version of (6) with the additional trust region constraints (12);
    if *the model is infeasible* **then**
        Reinitialize variables (apart from C and $\boldsymbol{\theta}$) with a single batch support vector learning;
    else
        $h \leftarrow$ measure of constraint violation;
        if (h, obj) *is acceptable to the filter* **then**
            Update linearly the optimization variables;
            $\rho \leftarrow 2\rho$;
            Update the filter;
            if N *iterations elapsed from the last swap* **then**
                Swap n points between $\mathcal{V}$ and $\mathcal{T}$ according to algorithm 1;
        else
            $\rho \leftarrow \frac{\rho}{2}$;

Algorithm 2: kernel learning algorithm.

scale of the signal but non-stationary at the scale of the dataset (considering that the dataset comes from the segmentation of a single long time signal).



(a) Pattern 1. (b) Pattern 2.

Fig. 1: Generative patterns of the toy problem. Each class of the dataset is built upon a pattern, which is randomly shifted and added to a colored noise.

In this experiment, we use the kernel described in section 2.3 in unison with 2 filters of length 32 and 4 segments in the pooling ($w = 32$). The confronted methods are i) the one presented in this paper[1] ("Our method with swap"), ii) the same without the swap heuristic ("Our method without swap"), iii) the common approach (3) solved with a projected gradient descent ("Large margin") and iv) a support vector machine with a filter bank kernel with two bandpass filters on the generative frequencies of the patterns ("Oracle"). The oracle somehow indicates the best results that can be achieved by learning methods in this context. For the bilevel approaches (with and without swap), 60% of the training dataset is used to actually train the classifier, and 40% to validate it. In the last two techniques, the unlearned parameters (respectively C and (C, γ)) are determined by a 5-fold cross-validation strategy. For bilevel and large margin approaches, C and γ are initialized to 1, and the filter bank to the simple low-pass $[\frac{1}{2}, \frac{1}{2}, 0, \ldots, 0]^T$ and high-pass $[\frac{1}{2}, -\frac{1}{2}, 0, \ldots, 0]^T$ filters. All the methods are learned with a small random training dataset (80 signals) and evaluated on another random test set. These steps are repeated 10 times so as to produce statistics.

The average classification accuracies on figure 2 show that our method is close to the oracle for medium and high signal-to-noise ratio (SNR) and performs better than the other approaches. Learning the parameters by solving (3) seems to overfit at any level of noise. Note also that the single validation estimate overfits for medium SNR. Figure 3 shows that our method is clearly faster than all the others (apart from the cross-validation of the oracle). Eventually, unreported results prove that, on this toy example, all the methods generalize as well when increasing the size of the training dataset.

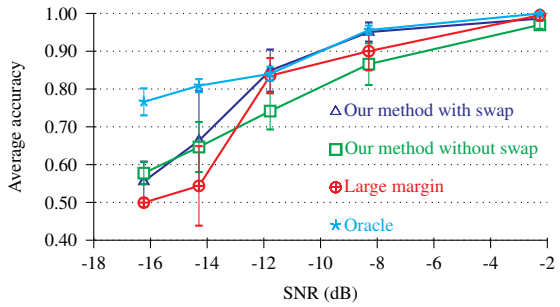


Fig. 2: Accuracy with respect to the SNR.

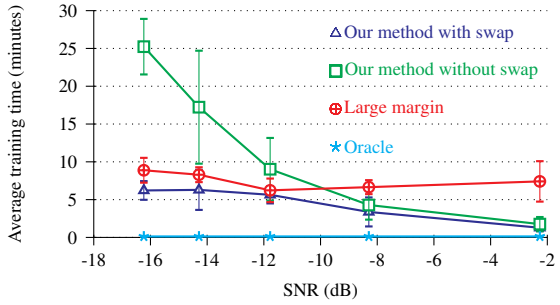


Fig. 3: Training time with respect to the SNR.

5. CONCLUDING REMARKS

On the premise that for small training datasets, kernel methods tend to overfit, we tried to minimize the most common generalization bound for support vector learning (*i.e.* the multiple validation estimate) taking into account the continuous nature of the parameters. The main contribution of this paper is to introduce a simple heuristic (made efficient thanks to recent advances in incremental support vector machines) in order to emulate several validation sets while considering only one at a time. The numerical experiments presented in this paper prove that for medium SNR, our approach is valuable compared to learning a kernel by only minimizing the structural risk or by using a single validation set.

[1] Like in (3), we also add the constraints $0 \leq \gamma$ and $\|\boldsymbol{h_1}\|_2^2 + \cdots + \|\boldsymbol{h_m}\|_2^2 \leq 1$ to keep the kernel positive-definite and to exclude degeneracies.

6. REFERENCES

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

[2] A. Rakotomamonjy, "Variable selection using SVM-based criteria," *Journal of Machine Learning Research*, vol. 3, pp. 1357–1370, 2003.

[3] R. Flamary, B. Labbé, and A. Rakotomamonjy, "Large margin filtering for signal sequence labeling," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2010.

[4] F. Yger and A. Rakotomamonjy, "Wavelet kernel learning," *Pattern Recognition*, vol. 44, pp. 2614–2629, 2011.

[5] G.R.G. Lanckriet, N. Cristianini, P.L. Bartlett, L. El Ghaoui, and M.I. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[6] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *Journal of Machine Learning Research*, vol. 9, pp. 2491–2521, 2008.

[7] C. Cortes, M. Mohri, and A. Rostamizadeh, "Algorithms for learning kernels based on centered alignment," *Journal of Machine Learning Research*, vol. 13, pp. 795–828, 2012.

[8] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine Learning*, vol. 46, pp. 131–159, 2002.

[9] S.S. Keerthi, V. Sindhwani, and O. Chapelle, "An efficient method for gradient-based adaptation of hyperparameters in SVM models," in *Advances in Neural Information Processing Systems (NIPS)*. 2006.

[10] C. Cortes, M. Kloft, and M. Mohri, "Learning kernels using local rademacher complexity," in *Advances in Neural Information Processing Systems (NIPS)*. 2013.

[11] K.P. Bennett, J. Hu, X. Ji, G. Kunapuli, and J.-S. Pang, "Model selection via bilevel optimization," in *International Joint Conference on Neural Networks*, 2006, pp. 1922–1929.

[12] G. Kunapuli, K.P. Bennett, Jing Hu, and Jong-Shi Pang, "Classification model selection via bilevel programming," *Optimization Methods and Software*, vol. 23, pp. 475–489, 2008.

[13] M. Karasuyama and I. Takeuchi, "Multiple incremental decremental learning of support vector machines," *IEEE Transactions on Neural Networks*, vol. 21, pp. 1048–1059, 2010.

[14] V.N. Vapnik, *Statistical learning theory*, Wiley, 1998.

[15] S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy, "SVM and kernel methods matlab toolbox," Perception Systèmes et Information, INSA de Rouen, Rouen, France, 2005.

[16] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27, 2011.

[17] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for SVMs," in *Advances in Neural Information Processing Systems (NIPS)*, 2000.

[18] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annals of Operations Research*, vol. 153, no. 1, pp. 235–256, 2007.

[19] R. Fletcher, N. I. M. Gould, S. Leyffer, P. L. Toint, and A. Wächter, "Global convergence of a trust-region SQP-filter algorithm for general nonlinear programming," *SIAM Journal on Optimization*, 2002.

[20] R. Fletcher, "The sequential quadratic programming method," in *Nonlinear Optimization*. Springer Berlin Heidelberg, 2010.