Solving ℓ^p-norm regularization with tensor kernels

Saverio Salzo
Istituto Italiano di Tecnologia

Johan A.K. Suykens
KU Leuven, ESAT-STADIUS

Lorenzo Rosasco
Università degli Studi di Genova

Abstract

In this paper, we discuss how a suitable family of tensor kernels can be used to efficiently solve nonparametric extensions of ℓ^p regularized learning methods. Our main contribution is proposing a fast dual algorithm, and showing that it allows to solve the problem efficiently. Our results contrast recent findings suggesting kernel methods cannot be extended beyond Hilbert setting. Numerical experiments confirm the effectiveness of the method.

1 Introduction

Kernel methods are classically formulated as a regularized empirical risk minimization and yields flexible and effective non-parametric models. However, they are restricted to ℓ^2-regularization. Indeed the so called *kernel trick* crucially relies on a scalar product structure (a Hilbert space). The basic tool of these methods is the kernel function which, evaluated at the training points, allows (a) to formulate a "dual" optimization problem, which is essentially quadratic and finite dimensional, and (b), through the solution of dual problem, to obtain an explicit linear representation of the solution of the original (primal) problem (*the representer theorem*) [18, 19]. This dual approach provides a feasible way to deal with non-parametric (infinite dimensional) models, and a possibly easier and more efficient algorithm to tackle the finite dimensional also.

It is well known that kernels for other norms can be defined [16, 20, 21], but recent results suggest that they are unpractical [17]. In particular, these kernels do not allows to properly express, in closed-form, the dual problem, making the kernel trick inapplicable. In this paper, we question this conclusion. We consider ℓ^p-regularization for $1 < p < 2$ and starting from [14] we illustrate how, for certain values of p, a class of tensor kernels make it possible to derive a dual problem that can be efficiently solved. Our main contribution is a dual algorithm, having fast convergence properties, that provides a way to overcome the well-known computational issues related to non-Hilbertian norms, and makes the kernel trick still viable. From the optimization point of view, the challenge is that some standard assumptions are not satisfied. Indeed the dual objective function lacks a global Lipschitz continuous gradient, since it incorporates a convex polynomial of degree strictly greater than 2. Moreover, depending on the choice of the loss, constraints may be present. Considering all these aspects, the proposed algorithm is a dual proximal gradient method with linesearch which in the case of the least square loss and logistic loss we prove to converge linearly. Numerical examples show the effectiveness of the proposed framework and the possible application for variable selection.

The rest of the paper is organized as follows. In section 2 we explain how tensor kernels arise in ℓ^p regularization learning problems and provide an efficient algorithm to solve such problems, which is the main contribution of the paper. In section 3 one finds the main elements of the theoretical analysis. Finally, section 4 contains the numerical experiments.

Notation. If $p > 1$, $q > 1$ is its conjugate exponent, i.e. $1/p + 1/q = 1$. Vectors are denoted by bold fonts and scalars by plain fonts. For every $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$, $\boldsymbol{x} \odot \boldsymbol{x}' \in \mathbb{R}^d$ and $\boldsymbol{x} \otimes \boldsymbol{x}' \in \mathbb{R}^{d\times d}$ are their Hadamard and tensor product respectively, and $\mathrm{sum}(\boldsymbol{x}) \in \mathbb{R}$ denotes the sum of the components of $\boldsymbol{x}$. If $\mathbb{K}$ is a countable set, we denote by $\ell^p(\mathbb{K})$ the space of p-summable sequences indexed in $\mathbb{K}$ with p-norm $\|\boldsymbol{w}\|_p = \big(\sum_{k\in\mathbb{K}} |w_k|^p\big)^{1/p}$. We define the *duality map* of $\ell^q(\mathbb{K})$ as $J_q\colon \ell^q(\mathbb{K}) \to \ell^p(\mathbb{K})$ with $J_q(\boldsymbol{u}) = (\mathrm{sign}(u_k)|u_k|^{q-1})_{k\in\mathbb{K}}$ [15].

2 Motivation and main contribution

First, we recall how kernel methods arise for ℓ^2-regularization. Next, we present the objective of this

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

study, i.e., an effective ℓ^p-norm regularized learning method. Based on [14], which showed that this method can be *kernelized* by an appropriate *tensor kernel*, we present a novel dual algorithm which uses the knowledge of the tensor kernel only and converges linearly.

2.1 Classical kernel methods

We begin with a look at a simple kernel method, that is, *kernel ridge regression*, and we highlight the role played by duality. Later, this will serve as a guide to generalize the theory to ℓ^p-regularization. Ridge regression is formulated as the following optimization problem

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \frac{\gamma}{2}\|\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y}\|_2^2+\frac{1}{2}\|\boldsymbol{w}\|_2^2, \tag{1}$$

where $\boldsymbol{X}\in\mathbb{R}^{n\times d}$ is the data matrix. This problem has a companion dual problem which is

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{1}{2}\|\boldsymbol{X}^*\boldsymbol{\alpha}\|_2^2+\frac{1}{2\gamma}\|\boldsymbol{\alpha}\|_2^2-\langle \boldsymbol{y},\boldsymbol{\alpha}\rangle, \tag{2}$$

where $\boldsymbol{X}^*$ is the transpose of $\boldsymbol{X}$. These two problems are indeed related: writing the optimality conditions for (1) and (2) one obtains

$$\begin{aligned}&\boldsymbol{X}^*(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y})+\gamma^{-1}\boldsymbol{w}=0 \qquad \text{and}\\ &\boldsymbol{X}\boldsymbol{X}^*\boldsymbol{\alpha}-\boldsymbol{y}+\gamma^{-1}\boldsymbol{\alpha}=0\end{aligned}$$

respectively; and hence it immediately follows that if $\bar{\boldsymbol{\alpha}}$ is the solution of (2), then

$$\bar{\boldsymbol{w}}=\boldsymbol{X}^*\bar{\boldsymbol{\alpha}}=\sum_{i=1}^n \bar{\alpha}_i\boldsymbol{x}_i \tag{3}$$

is the unique solution of (1). Equation (3) is the content of the so called *representer theorem* which ensures that the solution of a regularized regression problem can be written as a linear combination of the data points $\boldsymbol{x}_i\in\mathbb{R}^d$, $i=1,\cdots,n$. Moreover, for the linear estimator it holds

$$\langle\bar{\boldsymbol{w}},\boldsymbol{x}\rangle=\sum_{i=1}^n \bar{\alpha}_i\langle\boldsymbol{x}_i,\boldsymbol{x}\rangle=\sum_{i=1}^n \bar{\alpha}_i K(\boldsymbol{x}_i,\boldsymbol{x}), \tag{4}$$

where $K\colon\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}$ is the *linear kernel function* defined as $K(\boldsymbol{x},\boldsymbol{x}')=\langle\boldsymbol{x},\boldsymbol{x}'\rangle$. We note that, since $\boldsymbol{X}\boldsymbol{X}^*=(K(\boldsymbol{x}_i,\boldsymbol{x}_j))_{1\le i\le n,1\le j\le n}$, the dual problem (2) can also be written in terms of the linear kernel function, taking the form of the following quadratic optimization problem

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{1}{2}\sum_{i,j=1}^n K(\boldsymbol{x}_i,\boldsymbol{x}_j)\alpha_i\alpha_j+\frac{1}{2\gamma}\langle\boldsymbol{\alpha},\boldsymbol{\alpha}\rangle-\langle\boldsymbol{y},\boldsymbol{\alpha}\rangle. \tag{5}$$

So, summarizing, the dual problem (5) and the representation formulas (3)-(4) provide a way to solve the primal problem (1) and to evaluate the optimal linear estimator by relying on the knowledge of the linear kernel function only. This conclusion can then be extended to nonlinear regression models, by introducing general kernel functions defined as

$$K(\boldsymbol{x},\boldsymbol{x}')=\langle\Phi(\boldsymbol{x}),\Phi(\boldsymbol{x}')\rangle=\mathrm{sum}(\Phi(\boldsymbol{x})\odot\Phi(\boldsymbol{x}')), \tag{6}$$

for some nonlinear *feature map* $\Phi\colon\mathbb{R}^d\to\ell^2$. This is the so called *kernel trick* and it is at the basis of *kernel methods* in machine learning, allowing even to treat infinite dimensional (nonparametric) models. Kernels, defined by (6), can indeed be characterized as *positive definite functions*, in the sense that for every $n\in\mathbb{N}$, $(\boldsymbol{x}_i)_{1\le i\le n}\in(\mathbb{R}^d)^n$, and $\boldsymbol{\alpha}\in\mathbb{R}^n$, $\sum_{i,j=1}^n K(\boldsymbol{x}_i,\boldsymbol{x}_j)\alpha_i\alpha_j\ge 0$. Moreover, kernels define an associated function space which is a reproducing kernel Hilbert space. There are many significant examples of kernel functions and we cite among the other the Gaussian kernel $K(\boldsymbol{x},\boldsymbol{x}')=\exp(-\eta^{-2}\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2)$ and the polynomial kernel $K(\boldsymbol{x},\boldsymbol{x}')=\langle\boldsymbol{x},\boldsymbol{x}'\rangle^s$, describing the space of homogeneous polynomials of degree s. We note that the theory can be further generalized to handle more general loss functions, so to include classification problems too [18, 19].

2.2 Kernel methods beyond ℓ^2-regularization

In view of the discussion above, a natural question is whether kernel methods can be extended to other regularization terms. In particular ℓ^1-regularization would be important in view of its properties to provide sparse solutions. Unfortunately, in general ℓ^1-regularization methods cannot be *kernelized* (although they admit dual) [9, 11] and a useful representer theorem and definition of kernel can be obtained only under severe restrictions [16]. However, it was noted in [10] that ℓ^p-regularization can be seen as a proxy to ℓ^1 for suitable p. Moreover, it was recently shown in [14] that for certain values of $p\in\,]1,2[$ (arbitrarily close to 1), the ℓ^p-regularization method can indeed be *kernelized*, provided that a suitable definition of tensor kernel is introduced. Here we recall the theory in [14] for a simple model in order to make it more transparent. Thus, in analogy to section 2.1, we consider the problem

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \frac{\gamma}{2}\|\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y}\|_2^2+\frac{1}{p}\|\boldsymbol{w}\|_p^p:=F(\boldsymbol{w}), \tag{7}$$

where $1<p<2$. In this case the dual problem is

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^d} \frac{1}{q}\|\boldsymbol{X}^*\boldsymbol{\alpha}\|_q^q+\frac{1}{2\gamma}\|\boldsymbol{\alpha}\|_2^2-\langle\boldsymbol{y},\boldsymbol{\alpha}\rangle:=\Lambda(\boldsymbol{\alpha}), \tag{8}$$

where q is the conjugate exponent of p (that is $1/p+1/q=1$). Now, following the same argument as in

section 2.1, we write the optimality conditions of the two problems. Then we have

$$\begin{aligned} &\boldsymbol{X}^*(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y})+\gamma^{-1}J_p(\boldsymbol{w})=0 \quad \text{and} \\ &\boldsymbol{X}J_q(\boldsymbol{X}^*\boldsymbol{\alpha})-\boldsymbol{y}+\gamma^{-1}\boldsymbol{\alpha}=0, \end{aligned} \tag{9}$$

where $J_p\colon \mathbb{R}^d \to \mathbb{R}^d$ and $J_q\colon \mathbb{R}^d \to \mathbb{R}^d$ are the gradients of $(1/p)\|\cdot\|_p^p$ and $(1/q)\|\cdot\|_q^q$ respectively (they are the duality maps). Thus, multiplying by $\boldsymbol{X}^*$ the second equation in (9) and taking into account that $J_p \circ J_q = \mathrm{Id}$, it follows that if $\bar{\boldsymbol{\alpha}}$ is the solution of (8), then $\bar{\boldsymbol{w}} = J_q(\boldsymbol{X}^*\bar{\boldsymbol{\alpha}})$ is the solution of (7). So, in this case the *representer theorem* becomes

$$\bar{\boldsymbol{w}} = J_q(\boldsymbol{X}^*\bar{\boldsymbol{\alpha}}) = J_q\Big(\sum_{i=1}^n \bar{\alpha}_i \boldsymbol{x}_i\Big). \tag{10}$$

We remark that, in contrast to the ℓ^2 case, the above representation is nonlinear in the α_i's, because of the presence of the nonlinear map J_q. Indeed this map acts component-wise as the derivative of $(1/q)|\cdot|^q$, i.e., $\mathrm{sign}(\cdot)|\cdot|^{q-1}$. Therefore, at first sight it is not clear how to define an appropriate kernel function that can represent the estimator $\langle \bar{\boldsymbol{w}}, \boldsymbol{x}\rangle$ in analogy to (4), and make the kernel trick still successful. So, it comes as a surprise that this is possible if one makes the following assumption [14]

$$\boxed{q \text{ is an even integer and } q \geq 2.} \tag{11}$$

Indeed in that case, for every $\boldsymbol{u} \in \mathbb{R}^d$, $J_q(\boldsymbol{u}) = \big(\mathrm{sign}(u_j)|u_j|^{q-1}\big)_{1\leq j\leq d} = (u_j^{q-1})_{1\leq j\leq d}$, and hence, using (10), we have

$$\begin{aligned} \langle \bar{\boldsymbol{w}}, \boldsymbol{x}\rangle &= \sum_{j=1}^d \Big(\sum_{i=1}^n \bar{\alpha}_i x_{i,j}\Big)^{q-1} x_j \\ &= \sum_{j=1}^d \sum_{i_1,\dots,i_{q-1}=1}^n x_{i_1,j}\cdots x_{i_{q-1},j} x_j \bar{\alpha}_{i_1}\cdots\bar{\alpha}_{i_{q-1}}, \end{aligned} \tag{12}$$

where we could expand the power of the summation in a multilinear form since q is an integer. Therefore, we are defining the *linear tensor kernel function* K as

$$\begin{aligned} &K\colon \overbrace{\mathbb{R}^d \times \cdots \times \mathbb{R}^d}^{q \text{ times}} \to \mathbb{R}, \\ &K(\boldsymbol{x}'_1, \cdots, \boldsymbol{x}'_q) = \sum_{j=1}^d x'_{1,j}\cdots x'_{q,j} \\ &\qquad = \mathrm{sum}(\boldsymbol{x}'_1 \odot \cdots \odot \boldsymbol{x}'_q), \end{aligned} \tag{13}$$

so that, (12) turns to

$$\langle \bar{\boldsymbol{w}}, \boldsymbol{x}\rangle = \sum_{i_1,\dots,i_{q-1}=1}^n K(\boldsymbol{x}_{i_1}, \cdots, \boldsymbol{x}_{i_{q-1}}, \boldsymbol{x})\bar{\alpha}_{i_1}\cdots\bar{\alpha}_{i_{q-1}}. \tag{14}$$

Comparing (13) and (6) we recognize that we may interpret the tensor kernel as a kind of group-wise similarity measure in the input space. Moreover, since q is even,

$$\begin{aligned} \|X^*\boldsymbol{\alpha}\|_q^q &= \sum_{j=1}^d \Big(\sum_{i=1}^n \alpha_i x_{i,j}\Big)^q \\ &= \sum_{j=1}^d \sum_{i_1,\dots,i_q=1}^n x_{i_1,j}\cdots x_{i_q,j}\alpha_{i_1}\cdots\alpha_{i_q} \end{aligned}$$

and hence, by exchanging the two summations above, the dual problem (8) becomes

$$\begin{aligned} \min_{\boldsymbol{\alpha}\in\mathbb{R}^d} \frac{1}{q}\sum_{i_1,\dots,i_q=1}^n K(\boldsymbol{x}_{i_1},\dots,\boldsymbol{x}_{i_q})\alpha_{i_1}\cdots\alpha_{i_q} \\ + \frac{1}{2\gamma}\|\boldsymbol{\alpha}\|^2 - \langle \boldsymbol{y}, \boldsymbol{\alpha}\rangle. \end{aligned} \tag{15}$$

We see now that, instead of the quadratic problem (5) we have a convex polynomial optimization problem of degree q.[1] The introduction of the tensor kernel (13) allows to parallel the ℓ^2 case, in the sense that the dual problem (15) and formula (14) provide the solution of the regression problem (7). Once again, the method can be extended to general feature maps $\Phi\colon \mathbb{R}^d \to \ell^q(\mathbb{K})$, $\Phi(\boldsymbol{x}) = (\phi_k(\boldsymbol{x}))_{k\in\mathbb{K}}$, with $\mathbb{K}$ a countable set, provided that, in the definition of K, $\boldsymbol{x}_i$ is replaced by $\Phi(\boldsymbol{x}_i)$. Thus, a general *tensor kernel* is defined as

$$\begin{aligned} K(\boldsymbol{x}'_1, \cdots, \boldsymbol{x}'_q) &= \sum_{k\in\mathbb{K}} \phi_k(\boldsymbol{x}'_1)\cdots\phi_k(\boldsymbol{x}'_q) \\ &= \mathrm{sum}(\Phi(\boldsymbol{x}'_1) \odot \cdots \odot \Phi(\boldsymbol{x}'_q)). \end{aligned} \tag{16}$$

It is easy to show that tensor kernels are still symmetric and positive definite, in the sense that

- $\forall\, \boldsymbol{x}'_1, \dots, \boldsymbol{x}'_q \in \mathbb{R}^d$, and every permutation σ of $\{1,\dots,q\}$, $K(\boldsymbol{x}'_{\sigma(1)}\dots\boldsymbol{x}'_{\sigma(q)}) = K(\boldsymbol{x}'_1,\dots\boldsymbol{x}'_q)$;
- for every $\boldsymbol{x}'_1, \dots, \boldsymbol{x}'_n \in \mathbb{R}^d$ and every $\boldsymbol{\alpha} \in \mathbb{R}^n$, $\sum_{i_1,\dots,i_q=1}^n K(\boldsymbol{x}'_{i_1},\dots,\boldsymbol{x}'_{i_q})\alpha_{i_1}\dots\alpha_{i_q} \geq 0$.[2]

These tensor kernels define an associated function space which is now a reproducing kernel Banach space (See Section A.3 in the supplementary material and [14, 20]). Moreover, reasoning as in (12), the following representation formula can be proved

$$\langle \bar{\boldsymbol{w}}, \Phi(\boldsymbol{x})\rangle = \sum_{i_1,\dots,i_{q-1}=1}^n K(\boldsymbol{x}_{i_1}, \cdots, \boldsymbol{x}_{i_{q-1}}, \boldsymbol{x})\bar{\alpha}_{i_1}\cdots\bar{\alpha}_{i_{q-1}}. \tag{17}$$

[1] The problem is convex since the first term in (15) is equal to $(1/q)\|X^*\boldsymbol{\alpha}\|_q^q$.

[2] However, it is not known whether a function $K\colon (\mathbb{R}^d)^q \to \mathbb{R}$ satisfying the two properties above can be written as in (16) for some feature map $\Phi\colon \mathbb{R}^d \to \ell^q(\mathbb{K})$.

Finally, there do exist cases in which tensor kernel functions can be computed without knowing the feature map Φ itself. The following *polynomial* and *exponential* tensor kernels are examples of such cases (but, there are others in the class of power series tensor kernels [14]).

Polynomial tensor kernel of degree $s \in \mathbb{N}$**,** $s \geq 1$:

$$K(\boldsymbol{x}'_1, \dots, \boldsymbol{x}'_q) = \Big(\sum_{j=1}^{d} x'_{1,j} \cdots x'_{q,j} \Big)^s = \big(\mathrm{sum}(\boldsymbol{x}'_1 \odot \cdots \odot \boldsymbol{x}'_q) \big)^s.$$

It describes the space of homogeneous polynomials in d real variables of degree s. This corresponds to a finite dimensional model for which $\mathbb{K} = \big\{ k \in \mathbb{N}^d \,\big|\, \sum_{j=1}^d k_j = s \big\}$ and, for every $k \in \mathbb{N}^d$, $\phi_k(\boldsymbol{x}) = \big(s!/(k_1! \cdots k_d!) \big)^{1/q} \boldsymbol{x}^k$, that is $(\phi_k)_{k \in \mathbb{K}}$ is the basis of all possible monomials in d variables of degree s and the norm of a polynomial function $f = \sum_{k \in \mathbb{K}} w_k \phi_k$ is $\|\boldsymbol{w}\|_p^p = \sum_{k \in \mathbb{K}} |w_k|^p$.

Exponential tensor kernel :

$$K(\boldsymbol{x}'_1, \dots, \boldsymbol{x}'_q) = \prod_{j=1}^{d} e^{x'_{1,j} \cdots x'_{q,j}} = e^{\mathrm{sum}(\boldsymbol{x}'_1 \odot \cdots \odot \boldsymbol{x}'_q)}.$$

This kernel provides an example of an infinite dimensional model, where, $\mathbb{K} = \mathbb{N}^d$ and, for every $k \in \mathbb{N}^d$, the k-th component of the feature map is $\phi_k(\boldsymbol{x}) = \big(1/\prod_{j=1}^d k_j! \big)^{1/q} \boldsymbol{x}^k$.

2.3 A dual algorithm

In this section we present the main contribution of this paper which is an algorithm for solving the problem

$$\min_{\boldsymbol{w} \in \ell^p(\mathbb{K})} \gamma \sum_{i=1}^{n} \big(y_i - \langle \Phi(\boldsymbol{x}_i), \boldsymbol{w} \rangle \big)^2 + \frac{1}{p} \|\boldsymbol{w}\|_p^p := F(\boldsymbol{w}), \quad (18)$$

where $p = q/(q-1)$ with q an even integer (strictly) grater than 2, $\gamma > 0$, $\Phi \colon \mathcal{X} \to \ell^q(\mathbb{K})$ is the feature map, $\mathbb{K}$ is a countable set, and $(\boldsymbol{x}_i, y_i)_{1 \leq i \leq n} \in (\mathcal{X} \times \mathcal{Y})^n$ is the training set. Note that (18) reduces to (7) if $\mathcal{X} = \mathbb{R}^d$, $\mathbb{K} = \{1, \dots, d\}$ and Φ is the identity map. The proposed algorithm is based on the minimization of the dual problem (15), where K is defined as in (16). This method has two significant characteristics: first, it is entirely formulated in terms of the tensor kernel function, therefore it can also cope with non-parametric (infinite dimensional) tensor kernels, e.g, the exponential-tensor kernel; second, it provides fast convergence. From the optimization viewpoint, we observe that the objective functions in (18) and (15) are smooth. However, none of the two has Lipschitz continuous gradient, since in (18) $1 < p < 2$ and in (15) the first term is a convex polynomial of degree $q > 2$. This poses an issue since most gradient algorithms requires Lipschitz continuous gradient to achieve convergence [2, 7, 8]. Relaxing this assumption for the more general proximal gradient algorithm has been the objective of a number of recent works [3, 4, 13] that introduce suitable linesearch procedures to determine the gradient stepsizes. In light of these studies, we present a dual gradient descent algorithm with a backtracking linesearch procedure and we prove that, by exploiting the strong convexity of the dual objective function and the dual-primal link, the corresponding primal iterates converge linearly to the solution of (18).

To simplify the exposition we treat here the case $q = 4$, that is $p = 4/3$. Since the *Gram tensor* $\mathsf{K} = (K(\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \boldsymbol{x}_{i_3}, \boldsymbol{x}_{i_4}))_{i \in \{1, \dots n\}^4}$ is of order 4, it can be viewed as a $n^2 \times n^2$ symmetric matrix: using a MATLAB-like notation, we define $[\mathsf{K}] = \mathrm{reshape}(\mathsf{K}, n^2, n^2)$. Likewise, for a $n \times n$ matrix $\boldsymbol{B}$, we set $[\boldsymbol{B}] = \mathrm{reshape}(\boldsymbol{B}, n^2, 1)$ for its vectorization. Then, the dual problem (15) can be equivalently written as

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^d} \frac{1}{q} \langle [\boldsymbol{\alpha} \otimes \boldsymbol{\alpha}], [\mathsf{K}][\boldsymbol{\alpha} \otimes \boldsymbol{\alpha}] \rangle + \frac{1}{2\gamma} \|\boldsymbol{\alpha}\|^2 - \langle \boldsymbol{y}, \boldsymbol{\alpha} \rangle := \Lambda(\boldsymbol{\alpha}). \quad (19)$$

The proposed dual algorithm is detailed below.

Algorithm 2.1. Let $\boldsymbol{\alpha}_0 \in \mathbb{R}^n$, $\delta, \theta \in]0,1[$, and initialize the sequence $(\lambda_m)_{m \in \mathbb{N}}$ as the constant value $\bar{\lambda} \in]0, \gamma/(2(1-\delta))[$. Then, for every $m \in \mathbb{N}$,

$$\begin{aligned}
&\boldsymbol{\omega}_m = \mathrm{reshape}([\mathsf{K}][\boldsymbol{\alpha}_m \otimes \boldsymbol{\alpha}_m], n, n) \boldsymbol{\alpha}_m \\
&\text{(the gradient of the quartic part of } \Lambda) \\
&\nabla \Lambda(\boldsymbol{\alpha}_m) = \boldsymbol{\omega}_m - \boldsymbol{y} + \gamma^{-1} \boldsymbol{\alpha}_m \\
&\text{while } \Lambda(\boldsymbol{\alpha}_m) - \Lambda(\boldsymbol{\alpha}_m - \lambda_m \nabla \Lambda(\boldsymbol{\alpha}_m)) \\
&\qquad < \lambda_m (1-\delta) \|\nabla \Lambda(\boldsymbol{\alpha}_m)\|^2 \\
&\text{do} \\
&\quad \lambda_m := \theta \lambda_m \\
&\boldsymbol{\alpha}_{m+1} = (1 - \lambda_m \gamma^{-1}) \boldsymbol{\alpha}_m - \lambda_m (\boldsymbol{\omega}_m - \boldsymbol{y})
\end{aligned} \quad (20)$$

Remark 2.2. Algorithm 2.1 is given for $q = 4$. If q is an even integer greater than 4, then the leading term of Λ is a polynomial of degree q in the variables $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_m)$, and the formula for its gradient $\boldsymbol{\omega}_n$ at $\boldsymbol{\alpha}_m$, even if possibly more complicated, can be still expressed in term of the Gram tensor K.

Our main technical result is the following theorem studying the convergence of the above algorithm.

Theorem 2.3. *Let* $(\boldsymbol{\alpha}_m)_{m \in \mathbb{N}}$ *and* $(\lambda_m)_{m \in \mathbb{N}}$ *be generated by Algorithm 2.1. Then we have* $\inf_m \lambda_m > 0$ *and,*

for every $m \in \mathbb{N}$, *setting* $\boldsymbol{w}_m = J_q\big(\sum_{i=1}^n \alpha_{m,i}\Phi(\boldsymbol{x}_i)\big)$, *it holds*

$$\|\boldsymbol{w}_m - \bar{\boldsymbol{w}}\|_p^2 \leq \frac{\big[(2^p q)\big(\Lambda(\boldsymbol{\alpha}_0) + (\gamma/2)\|\boldsymbol{y}\|_2^2\big)\big]^{\frac{2-p}{p}}}{C_p} \cdot \left(1 - \frac{2}{\gamma}\lambda_m(1-\delta)\right)^m \big(\Lambda(\boldsymbol{\alpha}_0) - \min\Lambda\big),$$

for some constant $C_p > 0$, *depending only on* p, *which tends to zero as* $p \to 1$. *Therefore,* $\boldsymbol{w}_m$ *converges linearly to the solution* $\bar{\boldsymbol{w}}$ *of problem* (18).

Remark 2.4. An output $\boldsymbol{\alpha}_m$ of Algorithm 2.1 provides an estimator $\langle \boldsymbol{w}_m, \Phi(\cdot)\rangle$, that can be expressed in terms of the tensor kernel K through the equation

$$\langle \boldsymbol{w}_m, \Phi(\cdot)\rangle = \sum_{i_1,\dots,i_{q-1}=1}^{n} K(\boldsymbol{x}_{i_1}, \cdots, \boldsymbol{x}_{i_{q-1}}, \cdot)\alpha_{m,i_1}\cdots\alpha_{m,i_{q-1}}.$$

Indeed, this follows from recalling the definition of $\boldsymbol{w}_m$ in Theorem 2.3 and by reasoning as in (12).

Remark 2.5. If $p \in]1,2[$ is not of the form $p = q/(q-1)$ for some q as in (11), Theorem 2.3 remains valid provided that in Algorithm 2.1 $\boldsymbol{\omega}_m$ is computed directly in terms of the feature map Φ evaluated at the training points. Clearly, this case is feasible only if the feature map is finite dimensional, that is, if the index set $\mathbb{K}$ is finite.

In the following we discuss the most significant aspects of this dual approach.

Cost per iteration. The complexity of Algorithm 2.1 is mainly related to the computation of the gradient of the quartic form in (19), which, by exploiting the symmetries of $\boldsymbol{\alpha}_m \otimes \boldsymbol{\alpha}_m$ and $\mathbf{K}$, costs (approximatively) $n^2(n+1)^2/4$ multiplications. We remark that in the infinite dimensional case this algorithm is the only feasible approach to solve problem (18). However, even in the case $\mathrm{card}(\mathbb{K}) < +\infty$, e.g., for the linear or polynomial tensor kernel, the method may be convenient if $n \ll \mathrm{card}(\mathbb{K})$. Indeed a standard gradient-type algorithm on (18) costs $2n\mathrm{card}(\mathbb{K})$ multiplications ($2nd$ in case of (7)). Therefore, Algorithm 2.1 is recommended if $n(n+1)^2/8 \leq \mathrm{card}(\mathbb{K})$, that is

$$n \leq 2\big(\mathrm{card}(\mathbb{K})\big)^{1/3}. \tag{21}$$

We stress that Algorithm 2.1 has a cost per iteration that depends only on the size n of the data set, while any primal approach will depend on the size of $\mathbb{K}$. For instance, in the case of polynomial kernels of degree s, we have $\mathrm{card}(\mathbb{K}) = (d+s-1)\cdots d/s! \geq d^s/s!$, and this implies that the cost of a gradient algorithm on the primal problem grows exponentially with s. We also remark that building the Gram tensor $\mathbf{K}$ will further require $d \cdot n^2(n+1)^2/4$ multiplications (and $8 \cdot n^4/8$ bytes in space). However, the Gram tensor is computed once for all and in a validation procedure for the regularization parameter γ, it does not need to be recomputed every time.

Rate of convergence. As mentioned above our dual algorithm has linear convergence rate and can be applied for infinite dimensional kernels. We next discuss the comparison with primal approaches when the kernel is finite dimensional ($\mathrm{card}(\mathbb{K}) < +\infty$). The basic point is that primal approaches will allow only for sublinear rates. Indeed, since the objective function in (18) is the sum of two convex smooth functions, among the various algorithms, appropriate choices are (a) a pure gradient descent algorithm with linesearch (the gradient being that of F) and (b) a proximal gradient algorithm (possibly accelerated) with the prox of $(1/p)\|\cdot\|_p^p$. However, concerning (a) and according to [4, 13], the algorithm converges, but, since $1 < p < 2$, the full gradient of F is not even locally Lipschitz continuous, so, the gradient stepsizes may get arbitrarily close to zero, and ultimately the algorithm may exhibit very slow convergence with no explicit rate. Besides, regarding (b), the primal objective function in (18) is only uniformly convex on bounded sets. Therefore, standard convergence results [2, 6, 7] ensure only convergence of the iterates (without rate) and sublinear convergence rate for the objective values. On the other hand, regarding Algorithm 2.1, we observe that the constant C_p, in Theorem 2.3, approaches zero as $p \to 1$, so when p is close to 1 the linear convergence rate for the $\boldsymbol{w}_m$'s may degrade. In the numerical experiments, we confirm the above theoretical behaviors: the dual algorithm often converges in a few iterations (of the order of 20), whereas a direct gradient descent method (with linesearch or of proximal-type) on the primal problem may require thousands of iterations to reach the same precision.

Dealing with general convex loss. Above, we considered, for the sake of simplicity, the least squares loss. However, the proposed dual approach can be generalized to all other convex loss functions commonly used in machine learning: the *logistic loss* and the *hinge loss* for classification and the L^1*-loss*, and the *Vapnik-*ε*-insensitive loss* for regression. In these cases the dual objective function is composed of the same leading polynomial form as in (15), which has locally Lipschitz continuous gradient, and of a possibly nonsmooth (convex) function, having however a closed-form proximity operator (see Example B.2 in the supplementary material). Therefore, according to [13], for

general convex losses, instead of Algorithm 2.1 we use a proximal gradient algorithm with linesearch achieving linear convergence or sublinear convergence depending on the fact that the dual objective function is strongly convex or not. In this respect we note that we have linear convergence for the logistic loss and sublinear convergence for the ε-insensitive loss and the hinge loss. This extension is treated in the next section.

3 The main elements of the theoretical analysis

In this section we further develop the discussion of the previous section and provide the theoretical grounds for the dual approach to ℓ^p-norm regularized learning problems. The emphasis here is on the duality theory rather than on the tensor kernels. The results are presented for general loss function and any real parameter $p > 1$.

The most general formulation of our objective is as follows,

$$\min_{\boldsymbol{w}\in\ell^p(\mathbb{K})} \gamma\sum_{i=1}^{n} L(y_i, \langle\Phi(\boldsymbol{x}_i), \boldsymbol{w}\rangle) + \frac{1}{p}\|\boldsymbol{w}\|_p^p := F(\boldsymbol{w}), \tag{22}$$

where $p > 1, \gamma > 0$, $\Phi\colon \mathcal{X} \to \ell^q(\mathbb{K})$ is the feature map, $(\boldsymbol{x}_i, y_i)_{1\le i\le n} \in (\mathcal{X}\times\mathcal{Y})^n$ is the training set, and $L\colon \mathcal{Y}\times\mathbb{R}\to\mathbb{R}$ is a loss function which is convex in the second variable. We define the linear *feature operator*

$$\Phi_n\colon \ell^p(\mathbb{K}) \to \mathbb{R}^n, \quad \Phi_n\boldsymbol{w} = \big(\langle\Phi(\boldsymbol{x}_i), \boldsymbol{w}\rangle\big)_{1\le i\le n}. \tag{23}$$

Then its adjoint is $\Phi_n^*\colon \mathbb{R}^n \to \ell^q(\mathbb{K})$, $\Phi_n^*\boldsymbol{\alpha} = \sum_{i=1}^n \alpha_i\Phi(\boldsymbol{x}_i)$. Duality is based on the following.

Theorem 3.1. *The dual problem of* (22) *is*

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{1}{q}\|\Phi_n^*\boldsymbol{\alpha}\|_q^q + \gamma\sum_{i=1}^{n} L^*\Big(y_i, -\frac{\alpha_i}{\gamma}\Big) := \Lambda(\boldsymbol{\alpha}), \tag{24}$$

where $L^*(y_i,\cdot)$ is the Fenchel conjugate of $L(y_i,\cdot)$. Moreover, (i) the primal problem has a unique solution, the dual problem has solutions and $\min F = -\min\Lambda$ (strong duality holds); and (ii) the solutions $(\bar{\boldsymbol{w}}, \bar{\boldsymbol{\alpha}})$ of the primal and dual problems are characterized by the following KKT conditions

$$\begin{cases} \bar{\boldsymbol{w}} = J_q(\Phi_n^*\bar{\boldsymbol{\alpha}}), \\ \forall\, i\in\{1,\dots,n\} \quad -\frac{\alpha_i}{\gamma} \in \partial L(y_i, \langle\Phi(\boldsymbol{x}_i), \bar{\boldsymbol{w}}\rangle), \end{cases} \tag{25}$$

where $\partial L(y_i,\cdot)$ is the subdifferential of $L(y_i,\cdot)$.

All the losses commonly used in machine learning admit explicit Fenchel conjugates and we refer to the supplementary material for explicit examples. The connection between the primal and dual problem is further deepened in the following result.

Proposition 3.2. *Let $\bar{\boldsymbol{\alpha}} \in \mathbb{R}^n$ be a solution of the dual problem* (24) *and let $\bar{\boldsymbol{w}} = J_q\big(\Phi_n^*\bar{\boldsymbol{\alpha}}\big)$ be the solution of the primal problem* (22)*. Let $\boldsymbol{\alpha} \in \mathbb{R}^n$ and set $\boldsymbol{w} = J_q\big(\Phi_n^*\boldsymbol{\alpha}\big)$. Then*

$$\Lambda(\boldsymbol{\alpha}) - \min\Lambda \ge \frac{C_p}{\big[(2^pq)\big(\Lambda(\boldsymbol{\alpha}) + \gamma\|\boldsymbol{\xi}\|_1\big)\big]^{(2-p)/p}}\|\boldsymbol{w} - \bar{\boldsymbol{w}}\|_p^2, \tag{26}$$

where, for every $i = 1,\dots,n$, $\xi_i = \inf L^*(y_i,\cdot)$ and $C_p > 0$ is a constant that depends only on p.

The above proposition ensures that if an algorithm generates a sequence $(\boldsymbol{\alpha}_m)_{m\in\mathbb{N}}$ that is minimizing for the dual problem (24), i.e., $\Lambda(\boldsymbol{\alpha}_m) \to \min\Lambda$, then the sequence defined by $\boldsymbol{w}_m = J_q(\Phi_n^*\boldsymbol{\alpha}_m)$, $m \in \mathbb{N}$, converges to the solution of the primal problem.

Now, for the most significant losses L in machine learning (see Example B.2 in the supplementary material), the dual problem (24) has the following form

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \varphi_1(\boldsymbol{\alpha}) + \varphi_2(\boldsymbol{\alpha}) = \Lambda(\boldsymbol{\alpha}), \tag{27}$$

where $\varphi_1\colon \mathbb{R}^n \to \mathbb{R}$ is convex and smooth with locally Lipschitz continuous gradient (φ_1 will include the term $(1/q)\|\Phi_n^*\alpha\|_q^q$) and $\varphi_2\colon \mathbb{R}^n \to \mathbb{R}\cup\{+\infty\}$ is proper, lower semicontinuous, convex, and admitting a closed-form proximity operator. So, the form (24) is amenable by the proximal gradient algorithm with linesearch studied in [13], which, referring to (27), takes the following form.

Algorithm 3.3. Let $\delta \in]0,1[$, $\bar\lambda > 0$, and let $\theta \in]0,1[$. Let $\boldsymbol{\alpha}_0 \in \mathbb{R}^n$ and define, for every $m \in \mathbb{N}$,

$$\boldsymbol{\alpha}_{m+1} = \operatorname{prox}_{\lambda_m\varphi_2}(\boldsymbol{\alpha}_m - \lambda_m\nabla\varphi_1(\boldsymbol{\alpha}_m)), \tag{28}$$

where $\lambda_m = \bar\lambda\theta^{j_m}$ and j_m is the minimum of the $j \in \mathbb{N}$ such that $\hat{\boldsymbol{\alpha}}_m(j) := \operatorname{prox}_{\lambda_m\varphi_2}(\boldsymbol{\alpha}_m - \bar\lambda\theta^j\nabla\varphi_1(\boldsymbol{\alpha}_m))$ satisfies

$$\varphi_1\big(\hat{\boldsymbol{\alpha}}_m(j)\big) - \varphi_1\big(\boldsymbol{\alpha}_m\big) - \langle\hat{\boldsymbol{\alpha}}_m(j) - \boldsymbol{\alpha}_m, \nabla\varphi_1(\boldsymbol{\alpha}_m)\rangle \le \delta/(\bar\lambda\theta^j)\|\hat{\boldsymbol{\alpha}}_m(j) - \boldsymbol{\alpha}_m\|^2.$$

Remark 3.4. In contrast to Algorithm 2.1, Algorithm 3.3 provides rather a general algorithm where φ_1 and φ_2 are set depending on the choice of the different losses.

Remark 3.5. If $p = q/(q-1)$ and q satisfies (11), then the computation of $\nabla\varphi_1(\boldsymbol{\alpha})$ in Algorithm 3.3 can be performed in term of the Gram tensor K (for instance, if $q = 4$ the gradient of the quartic part of φ_1 is as in the first line of Algorithm 2.1). Moreover, if in addition L is the square loss, then Λ is as in (19) and one can take $\varphi_1 = \Lambda$ and $\varphi_2 = 0$; and hence Algorithm 3.3 reduces to Algorithm 2.1.

Table 1: Convergence rates

	Number of iterations (rel. precision 10^{-8})			
Algorithm	$p = 4/3$	$p = 5/4$	$p = 1.1$	$p = 1.05$
dual GD + linesearch	12(5)	15(4)	63(22)	258(55)
primal GD + linesearch	> 5000	> 5000	> 5000	> 5000
primal FISTA	1158	1542	—	—

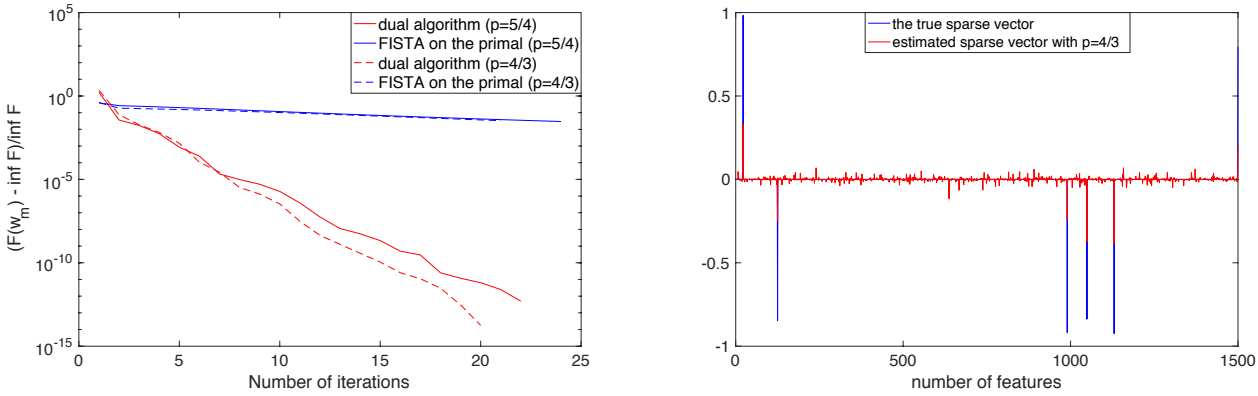

Figure 1: Left. Convergence rates: dual algorithm vs FISTA on the primal. Right. True and estimated sparse vectors for a linear tensor kernel: $p = 4/3$, $n = 85$, $d = 1500$, and 6 relevant features.

The convergence properties of Algorithm 3.3 are given in the following theorem, which, as opposed to Theorem 2.3, is valid for general loss and any $p \in]1, 2]$.

Theorem 3.6. *Let $p \in]1, 2]$. Define $(\boldsymbol{\alpha}_m)_{m\in\mathbb{N}}$ and $(\lambda_m)_{m\in\mathbb{N}}$ as in Algorithm 3.3. Then, $\inf_m \lambda_m > 0$ and, for every $m \in \mathbb{N}$, setting $\boldsymbol{w}_m = J_q(\Phi_n^* \boldsymbol{\alpha}_m)$, it holds*

$$\|\boldsymbol{w}_m - \bar{\boldsymbol{w}}\|_p \leq o(1/\sqrt{m}).$$

Moreover, if Λ is strongly convex (which occurs for the least square loss and the logistic loss), then $\boldsymbol{w}_m$ converges linearly to $\bar{\boldsymbol{w}}$.

4 Numerical Experiments

In this section we perform experiments on simulated data. Also, we provide experiments on a real data set the supplementary material.

The experiments on simulated data assesses the following three points.[3]

Dual vs primal approach (without tensor kernels). We considered problem (7) with different choices of p (not necessarily with q even integer). The purpose is to compare a dual approach against a primal approach per se, thus without considering the tensor kernel function — after all the dual problem (8) is smooth whatever q is. Algorithm 2.1 is therefore modified in such a way that the gradient of the dual term $(1/q)\|\boldsymbol{X}^*\boldsymbol{\alpha}\|_q^q$ is computed directly as $\boldsymbol{X} J_q(\boldsymbol{X}^*\boldsymbol{\alpha})$.[4] For the primal approaches we considered two algorithms: (a) the gradient descent method with linesearch and (b) the FISTA algorithm [2], but with $p \in \{4/3, 5/4\}$, since they are the only cases in which the proximity operator of $(1/p)\|\cdot\|_p^p$ can be computed explicitly [1]. We generated a matrix $\boldsymbol{X}$ according to a normal distribution, a sparse vector $\boldsymbol{w}_\star$, (where the location of the nonzero coefficients was chosen randomly), a normal distributed noise vector $\boldsymbol{\varepsilon}$, and we defined

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}_\star + \sigma\boldsymbol{\varepsilon}, \qquad \sigma = 5 \cdot 10^{-2}.$$

We chose $n = 200$, $d = 10^5$ and 10 relevant features. The regularization parameter was set to $\gamma = 10$, so to achieve a reconstruction error of the order of the noise. Table 1 and Figure 1(Left.) clearly show that the dual approach significantly outperforms the two primal approaches.[5]

[3] All the numerical experiments have been performed in MATLAB® environment, on a MacBook laptop with Intel Core 2 Duo, 2 Ghz and 4 GB of RAM.

[4] Note that in this case the cost per iteration is essentially equal to that of the gradient descent in the primal.

[5] The optimal values were found by using the dual algorithm and checking that the duality gap was $< 10^{-14}$.

Table 2: The dual algorithm with and without tensor kernels

Algorithm	CPU time (sec)	iterations
build the Gram tensor K	2.73	—
dual GD + linesearch (with K)	2.49	29
dual GD + linesearch (without K)	9.87	28

Tensor kernels in the dual approach. This experiment considered the case treated in section 2.3, that is, $q = 4$ ($p = 4/3$), with the polynomial tensor kernel of degree 2, i.e.,

$$\begin{aligned} K(\boldsymbol{x}'_1,&\boldsymbol{x}'_2,\boldsymbol{x}'_3,\boldsymbol{x}'_4) \\ &= \big(\mathrm{sum}(\boldsymbol{x}'_1 \odot \boldsymbol{x}'_2 \odot \boldsymbol{x}'_3 \odot \boldsymbol{x}'_4)\big)^2 \\ &= \mathrm{sum}(\Phi(\boldsymbol{x}'_1) \odot \Phi(\boldsymbol{x}'_2) \odot \Phi(\boldsymbol{x}'_3) \odot \Phi(\boldsymbol{x}'_4)), \end{aligned}$$

where $\Phi(\boldsymbol{x}) = \big(x_1^2, \dots, x_d^2, \sqrt[4]{2}x_1x_2, \dots, \sqrt[4]{2}x_1x_d, \dots\big)$. The dimension of the feature space is $N = d(d+1)/2$. We generated $\boldsymbol{X}$, $\boldsymbol{w}_\star$, $\boldsymbol{\varepsilon}$ as in the previous case and, according to (23),[6] we defined

$$\Phi_n = \begin{bmatrix} \Phi(\boldsymbol{x}_1)^\top \\ \vdots \\ \Phi(\boldsymbol{x}_d)^\top \end{bmatrix} \in \mathbb{R}^{n\times N},\ \boldsymbol{y} = \Phi_n\boldsymbol{w}_\star + \sigma\boldsymbol{\varepsilon},\ \sigma = 5{\cdot}10^{-2}.$$

Then we aimed at solving problem (7) with $\boldsymbol{X}$ replaced by Φ_n. We examined a situation in which the computational cost per iteration of the dual algorithm is less than the corresponding primal, measuring the gain in CPU time. We set $n = 90$, $d = 650$ and 6 relevant features out of the total of $N = 211575$. With these figures, according to the discussion at the end of section 2.3, computing the gradient through the tensor kernel, as done in Algorithm 2.1, surely reduces the cost per iteration. Table 2 shows the CPU time required by the dual algorithm with and without using the tensor kernel.

Recovering the relevant features. The sparseness properties of an ℓ^p-regularization method were mentioned in [8] and later were studied further in [10], from a statistical viewpoint. In contrast to ℓ^1-regularization, the ℓ^p-regularization does not generally provide finite supported vectors, so sparseness here actually means *approximate sparsity* in the sense that the insignificant coefficients are shrunk and the relevant ones are highlighted. Our experiments confirm this property of ℓ^p regularization. Indeed in the setting described in the previous scenarios, the solution vector $\bar{\boldsymbol{w}}$ always exhibits spikes that corresponds to the non zero coefficients of $\boldsymbol{w}_\star$. Depending on the value of p, on the size n of the data set, and on the feature space dimension N, this phenomenon may be more or less notable, but in any case the vector $\bar{\boldsymbol{w}}$ either clearly reveals the hidden relevant features (see Figure 1(Right.)) or can be safely thresholded in order to discard most of the non-relevant features, and reduce the dimensionality of the problem of 1-2 orders of magnitude.

[6] In this case $\mathrm{card}(\mathbb{K}) = N$, so $\ell^p(\mathbb{K})$ can be identified with $\mathbb{R}^N$ and the linear map Φ_n can be thought as a $n \times N$ matrix.

5 Conclusions

In this paper we presented a novel and efficient kernel method for ℓ^p-norm regularized learning problems. The method assumes that $p = q/(q-1)$ with q an even integer grater than 2. In such case, we provided an algorithm which is based on the minimization of the dual problem and can be formulated in terms of a tensor kernel evaluated at the training points, avoiding the call of the feature map. Therefore, this provides the first viable solution to ℓ^p-type regularization in infinite dimensional spaces. Moreover, in finite dimension, the proposed approach compares favorably to other solutions in the regime of few sample and large number of variables, and q reasonably low. For example, our experiments show that if $q = 4$, the proposed method is practicable and provides an effective variables selection method and/or is able to discard most of the irrelevant features. We remark that, the complexity of the method depends only on the dataset size and does not depend on the dimension of the function space (e.g, the degree of the polynomial kernel). However, there are scenarios and values of q in which using tensor kernels may be cumbersome from the computational point of view, but this difficulty is common to other approaches to nonparametric sparsity and it is certainly a challenge that requires further study. Finally, the experiments are meant to provide a proof of concept for the proposed method and are the starting point for a more systematic empirical study that we defer to a future work.

Acknowledgements

Johan Suykens acknowledges support from the European Research Council ERC AdG A-DATADRIVE-B (290923), KU Leuven CoE PFV/10/002, FWO G.0377.12, G.088114N, IUAP P7/19 DYSCO.

References

[1] H.H Bauschke and P.L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces. 2nd Ed.* Springer, New York, 2017.

[2] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1):183–202, 2009.

[3] J.Y. Bello Cruz and T.T.A. Nghia. On the convergence of the proximal forward-backward splitting method with linesearch. arXiv:1603.05876, 2015.

[4] S. Bonettini, L. Loris, F. Porta, and M. Prato. Variable metric inexact line-search based methods for nonsmooth optimization. *SIAM J. Optim.*, 26(2):891–921, 2016.

[5] K. Bredies and D.A. Lorenz. Linear convergence of iterative soft-thresholding. *J. Fourier Anal. Appl.*, 14:813–837, 2008.

[6] A. Chambolle and Ch. Dossal. On the convergence of the iterates of the "fast iterative shrinkage/thresholding algorithm". *J Optim Theory Appl (2015)*, 166:968982, 2015.

[7] P.L. Combettes and V.R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Model. Simul.*, 4:1168–1200, 2005.

[8] I. Daubechies, M. Defrise, and De Mol C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. Pure Appl. Math.*, 57(11):1413–1457, 2004.

[9] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations.* Chapman and Hall/CRC, 2015.

[10] V. Koltchinskii. Sparsity in penalized empirical risk minimization. *Ann. Inst. Henri Poincaré Probab. Stat.*, 45:7–57, 2009.

[11] M.R. Osborne, B. Presnell, and B.A. Turnach. On the LASSO and its dual. *J. Comp. Graph. Stat.*, 9:319–337, 2000.

[12] L. Rosasco, S. Villa, S. Mosci, M. Santoro, and A. Verri. Nonparametric sparsity and regularization. *J. Mach. Learn. Res.*, 14:1665–1714, 2013.

[13] S. Salzo. The variable metric forward-backward splitting algorithm under mild differentiability assumptions. *SIAM J. Optim.*, 27(4):2153–2181, 2017.

[14] S. Salzo and J.A.K. Suykens. Generalized support vector regression: duality and tensor-kernel representation. arXiv:1603.05876:1–30, 2016.

[15] T. Schuster, B. Kaltenbacher, B. Hofmann, and K.S. Kazimierski. *Regularization Methods in Banach spaces.* De Gruyter, Berlin, 2012.

[16] G. Song, H. Zhang, and F.J. Hickernell. Reproducing kernel Banach spaces with ℓ^1 norm. *Appl. Comput. Harmon. Anal.*, 34:96–116, 2013.

[17] B.H. Sriperumbudur, K. Fukumizu, and G.R.G. Lanckriet. Learning in Hilbert vs. Banach spaces: a measure embedding viewpoint. In *Advances in Neural Information Processing Systems 24.*, 2011.

[18] I. Steinwart and A. Christmann. *Support Vector Machines.* Springer, New York, 2008.

[19] V.N. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[20] H. Zhang, Y. Xu, and H. Zhang. Reproducing kernel Banach spaces for machine learning. *J. Mach. Learn. Res.*, 10:2741–2775, 2009.

[21] H. Zhang and H. Zhang. Regularized learning in Banach spaces as an optimization problem: representer theorems. *J. Global Optim.*, 54:235–250, 2012.