Solving Structured Sparsity Regularization with Proximal Methods

Sofia Mosci[1], Lorenzo Rosasco[3,4], Matteo Santoro[1],
Alessandro Verri[1], and Silvia Villa[2]

[1] Università degli Studi di Genova - DISI
Via Dodecaneso 35, Genova, Italy
[2] Università degli Studi di Genova - DIMA
Via Dodecaneso 35, Genova, Italy
[3] Istituto Italiano di Tecnologia,
Via Morego, 30 16163 Genova, Italy
[4] CBCL, Massachusetts Institute of Technology
Cambridge, MA 02139 - USA

Abstract. Proximal methods have recently been shown to provide effective optimization procedures to solve the variational problems defining the ℓ_1 regularization algorithms. The goal of the paper is twofold. First we discuss how proximal methods can be applied to solve a large class of machine learning algorithms which can be seen as extensions of ℓ_1 regularization, namely structured sparsity regularization. For all these algorithms, it is possible to derive an optimization procedure which corresponds to an iterative projection algorithm. Second, we discuss the effect of a preconditioning of the optimization procedure achieved by adding a strictly convex functional to the objective function. Structured sparsity algorithms are usually based on minimizing a convex (not strictly convex) objective function and this might lead to undesired unstable behavior. We show that by perturbing the objective function by a small strictly convex term we often reduce substantially the number of required computations without affecting the prediction performance of the obtained solution.

1 Introduction

In this paper we show how proximal methods can be profitably used to study a variety of machine learning algorithms. Recently, methods such as the lasso [22] – based on ℓ_1 regularization – received considerable attention for their property of providing sparse solutions. Sparsity has become a popular way to deal with small samples of high dimensional data and, in a broad sense, refers to the possibility of writing the solution in terms of a few building blocks. The success of ℓ_1 regularization motivated exploring different kinds of sparsity enforcing penalties for linear models as well as kernel methods [13, 14, 18, 25–27]. A common feature of this class of penalties is that they can be often written as suitable sums of euclidean (or Hilbertian) norms.

J.L. Balcázar et al. (Eds.): ECML PKDD 2010, Part II, LNAI 6322, pp. 418–433, 2010.

On the other hand, proximal methods have recently been shown to provide effective optimization procedures to solve the variational problems defining the ℓ_1 regularization algorithms, see [3, 4, 6, 7, 19] and [11] in the specific context of machine learning. In the following we discuss how proximal methods can be applied to solve the class of machine learning algorithms which can be seen as extensions of ℓ_1 regularization, namely structured sparsity regularization. For all these algorithms, it is possible to derive an optimization procedure that corresponds to an efficient iterative projection algorithm which can be easily implemented. Depending on the considered learning algorithm, the projection can be either computed in closed form or approximated by another proximal algorithm. A second contribution of our work is to study the effect of a preconditioning of the optimization procedure achieved by adding a strictly convex functional to the objective function. Indeed, structured sparsity algorithms are usually based on minimizing a convex (not strictly convex) objective function and this might lead to undesired unstable behavior. We show that by perturbing the objective function with a small strictly convex term it is possible to reduce substantially the number of required computations without affecting the prediction property of the obtained solution.

The paper is organized as follows. In Section 2, we begin by setting the notation, necessary to state all the mathematical and algorithmic results presented in Section 3. In Section 4, in order to show the wide applicability of our work, we apply the results to several learning schemes, and in Section 5 we describe the experimental results. An extended version of this work can be found in [21], where the interested reader can find all the proofs and some more detailed discussions.

2 Setting and Assumptions

In this section we describe the setting of structured sparsity regularization, in which a central role is played by the following variational problem. Given a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$, and two fixed positive numbers τ and μ, we consider the problem of computing:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}} \mathcal{E}_{\tau,\mu}(f) = \operatorname*{argmin}_{f \in \mathcal{H}} \{F(f) + 2\tau J(f) + \mu \|f\|_{\mathcal{H}}^2\}, \tag{1}$$

where $F : \mathcal{H} \to \mathbb{R}$, $J :\to \mathbb{R} \cup \{+\infty\}$ represent the data and penalty terms, respectively, and $\mu \|f\|_{\mathcal{H}}^2$ is a perturbation discussed below. Note that the choice of RKHS recovers also the case of generalized linear models where f can be written as $f(x) = \sum_{j=1}^{d} \beta_j \psi_j(x)$ for a given dictionary $(\psi_j)_{j=1}^d$, as well as more general models (see Section 4). In the following, F is assumed to be differentiable and convex. In particular we are interested in the case where the first term is the empirical risk associated to a training set $\{(x_i, y_i)_{i=1}^n\} \subseteq (X \times [-C, C])^n$, and a cost function $\ell : \mathbb{R} \times [-C, C] \to \mathbb{R}^+$,

$$F(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i). \tag{2}$$

and specifically in the case where $\ell(y, f(x))$ is the square loss $(y - f(x))^2$. (other losses – e.g. the logistic loss – would also fit our framework).

In the following we require J to be lower semicontinuous, convex, and one-homogeneous, $J(\lambda f) = \lambda J(f)$, for all $f \in \mathcal{H}$ and $\lambda \in \mathbb{R}^+$. Indeed these technical assumptions are all satisfied by the vast majority of penalties commonly used in the recent literature of sparse learning. The main examples for the functional J are penalties which are sum of norms in distinct Hilbert spaces $(\mathcal{G}_k, \|\cdot\|_k)$:

$$J(f) = \sum_{k=1}^{M} ||\mathcal{J}_k(f)||_k, \tag{3}$$

where, for all k, $\mathcal{J}_k : \mathcal{H} \to \mathcal{G}_k$ is a bounded linear operator bounded from below. This class of penalties have recently received attention since they allow one to enforce more complex sparsity patterns than the simple ℓ_1 regularization [13, 26]. The regularization methods induced by the above penalties are often referred to as structured sparsity regularization algorithms.

Before describing how proximal methods can be used to compute the regularized solution of structured sparsity methods we note that in general, if we choose F, J as in (2) and (3), when $\mu = 0$, the functional (1) will be convex but not strictly convex. Then the regularized solution is in general not unique. On the other hand by setting $\mu > 0$, strict convexity, hence uniqueness of the solution, is guaranteed. As we discuss in the following, this can be seen as a preconditioning of the problem, and, if μ is small enough, one can see empirically that the solution does not change.

3 General Iterative Algorithm

In this section we describe the general iterative procedure for computing the solution f^* of the convex minimization problem (1).

Let K denote the subdifferential, $\partial J(0)$, of J at the origin, which is a convex and closed subset of $\mathcal{H}$. For any $\lambda \in \mathbb{R}^+$, we call $\pi_{\lambda K} : \mathcal{H} \to \mathcal{H}$ the projection on $\lambda K \subset \mathcal{H}$. The optimization scheme we derive is summarized in Algorithm 1, the parameter σ can be seen as a step-size, which choice is crucial to ensure convergence and is discussed in the following subsection. In general, approaches based on proximal methods decouple the contributions of the two functionals J

Algorithm 1. General Algorithm

Require: $\bar{f} \in \mathcal{H}$, $\sigma, \tau, \mu > 0$
Initialize: $f^0 = \bar{f}$
while `convergence not reached` **do**
$p := p + 1$

$$f^p = \left(I - \pi_{\frac{\tau}{\sigma}K}\right)\left(\left(1 - \frac{\mu}{\sigma}\right) f^{p-1} - \frac{1}{2\sigma}\nabla F(f^{p-1})\right) \tag{4}$$

end while
return f^p

and F, since, at each iteration, the projection $\pi_{\tau/\sigma K^-}$ which is entirely characterized by J – is applied to a term that depends only on F. The derivation of the above iterative procedure for a general non differentiable J relies on a well-known result in convex optimization (see [6] for details), that has been used in the context of supervised learning by [11]. We recall it for completeness and because it is illustrative for studying the effect of the perturbation term $\mu \|f\|^2_{\mathcal{H}}$ in the context of structured sparsity regularization.

Theorem 1. *Given $\tau, \mu > 0$, $F : \mathcal{H} \to \mathbb{R}$ convex and differentiable and $J : \mathcal{H} \to \mathbb{R} \cup \{+\infty\}$ lower semicontinuous and convex, for all $\sigma > 0$ the minimizer f^* of $\mathcal{E}_{\tau,\mu}$ is the unique fixed point of the map $\mathcal{T}_\sigma : \mathcal{H} \to \mathcal{H}$ defined by*

$$\mathcal{T}_\sigma(f) = \operatorname{prox}_{\frac{\tau}{\sigma}J}\left(\left(1 - \frac{\mu}{\sigma}\right) f - \frac{1}{2\sigma}\nabla F(f)\right),$$

where $\operatorname{prox}_{\frac{\tau}{\sigma}J}(f) = \operatorname{argmin}\left\{\frac{\tau}{\sigma}J(g) + \frac{1}{2}\|f - g\|^2\right\}$.

For suitable choices of σ the map $\mathcal{T}_\sigma$ is a contraction, thus convergence of the iteration is ensured by Banach fixed point theorem and convergence rates can be easily obtained- see next section. The case $\mu = 0$ already received a lot of attention, see for example [3, 4, 6, 7, 19] and references therein.

Here we are interested in the setting of supervised learning when the penalty term is one-homogeneous and, as said before, enforces some structured sparsity property. In [21] we show that such assumption guarantees that

$$\operatorname{prox}_{\frac{\tau}{\sigma}J} = \left(I - \pi_{\frac{\tau}{\sigma}K}\right).$$

In the following subsection we discuss the role of the perturbation term $\mu \|f\|^2_{\mathcal{H}}$.

3.1 Convergence and the Role of the Strictly Convex Perturbation

The effect of $\mu > 0$ is clear if we look at convergence rates for the map $\mathcal{T}_\sigma$. In fact it can be shown ([21]) that a suitable a priori choice of σ is given by

$$\sigma = \frac{1}{4}(aL_{max} + bL_{min}) + \mu,$$

where a and b denote the largest and smallest eigenvalues of the kernel matrix, $[\mathrm{K}]_{i,j} = k(x_i, x_j), i, j = 1, \ldots, n$, with k the kernel function of the RKHS $\mathcal{H}$, and $0 \leq L_{min} \leq \ell''(w, y) \leq L_{max}$, $\forall w \in \mathbb{R}, y \in Y$, where ℓ'' denotes the second derivative of ℓ with respect to w. With such a choice the convergence rate is linear, i.e.

$$\|f^* - f^p\| \leq \frac{L_\sigma^p}{1 - L_\sigma}\|f^1 - f^0\|, \qquad \text{with} \quad L_\sigma = \frac{aL_{max} - bL_{min}}{aL_{max} + bL_{min} + 4\mu}. \tag{5}$$

Typical examples of loss functions are the square loss and the exponential loss. In these cases suitable step sizes are $\sigma = \frac{1}{2}(a + b + 2\mu)$, and $\sigma = \frac{1}{4}(aC^2e^{C^2}) + \mu$, respectively.

The above result highlights the role of the μ-term, $\mu \|\cdot\|^2_{\mathcal{H}}$, as a natural preconditioning of the algorithm. In fact, in general, for a strictly convex F, if the

smallest eigenvalue of the second derivative is not uniformly bounded from below by a strictly positive constant, when $\mu = 0$, it might not be possible to choose σ so that $L_\sigma < 1$. One can also argue that, if μ is chosen small enough, the solution is expected not to change and in fact converges to a precise minimizer of $F + 2\tau J$. Infact, the quadratic term performs a further regularization that allows to select, as μ approaches 0, the minimizer of $F + 2\tau J$ having minimal norm (see for instance [10]).

3.2 Computing the Projection

In order to compute the proximity operator associated to a functional J as in (3), it is useful to define the operator $\mathcal{J} : \mathcal{H} \to \prod \mathcal{G}_k$ as $\mathcal{J}(f) = (\mathcal{J}_1(f), \ldots, \mathcal{J}_M(f))$. With this definition the projection of an element $f \in \mathcal{H}$ on the set $\lambda K := \lambda \partial J(0)$ is given by $\mathcal{J}^T \bar{v}$, where

$$\bar{v} \in \operatorname*{argmin}_{v \in \mathcal{G}} \|\mathcal{J}^* v - f\|_{\mathcal{H}}^2 + I_{\lambda B}(v), \quad \text{with } I_{\lambda B}(v) = \begin{cases} 0 & \text{if } \|v_k\|_k \leq \lambda \, \forall k \\ +\infty & \text{otherwise.} \end{cases} \tag{6}$$

The computation of the solution of the above equation is different in two distinct cases. In the first case $\mathcal{H}_k = \mathcal{G}_k, \mathcal{H} = \prod \mathcal{H}_k$, and $\mathcal{J}_k$ is the weighted projection operator on the k-th component, i.e. $\mathcal{J}_k(v) = v_k \in \mathcal{H}_k$ with $w_k > 0, \forall k$, and $\bar{v}$ can be computed exactly as $\bar{v} = \pi_{\lambda B}(f)$, where $\bar{v} = \pi_{\lambda B}(f)$ is simply the projection on the cartesian product of k balls of radius λw_k

$$(\pi_{\lambda B})_k(f_k) = \min \left\{ 1, \frac{\lambda w_k}{\|f_k\|_k} \right\} f_k.$$

In this case $(I - \pi_{\lambda K})$ coincides with the block-wise soft-thresholding operator:

$$(I - \pi_{\lambda K})_k(f_k) = (\|f_k\|_k - \lambda w_k)_+ f_k := S_\lambda(f_k) \tag{7}$$

which reduces to the well-known component-wise soft-thresholding operator when $\mathcal{H}_k = \mathbb{R}$ for all k.

On the other hand, when $\mathcal{J}$ is not a blockwise weighted projection operator, $\pi_{\lambda K}(f)$ cannot be computed in closed form. In this case we can again resort to proximal methods since equation (6) amounts to the minimization of a functional which is sum of a differential term $\|\mathcal{J}^* v - f\|_{\mathcal{H}}^2$ and a non-differential one $I_{\lambda B}(v)$. We can therefore apply Theorem 1 in order to compute $\bar{v}$ which is the fixed point of the map $\mathcal{T}_\eta$ defined as

$$\mathcal{T}_\eta(v) = \pi_{\lambda B}\left(v - (\eta)^{-1} \mathcal{J}(\mathcal{J}^* v - f)\right) \quad \text{for all } \eta > 0 \tag{8}$$

where $\operatorname{prox}_{I_{\lambda B}} = \pi_{\lambda B}$. We can therefore evaluate it iteratively as $v^q = \mathcal{T}_\eta(v^{q-1})$.

3.3 Some Relevant Algorithmic Issues

Adaptive Step-Size Choice. In the previous sections we proposed a general scheme as well as a parameter set-up ensuring convergence of the proposed

procedure. Here, we discuss some heuristics that were observed to consistently speed up the convergence of the iterative procedure. In particular, we mention the Barzilai-Borwein methods – see for example [15, 16, 23] for references. More precisely in the following we will consider

$$\sigma_p = \langle s^p, r^p \rangle / \|s^p\|^2, \qquad \text{or} \quad \sigma_t = \|r^p\|^2 / \langle s^p, r^p \rangle.$$

where $s^p = f^p - f^{p-1}$ and $r^p = \nabla F(f^p) - \nabla F(f^{p-1})$.

Continuation Strategies and Regularization Path. Finally, we recall the *continuation* strategy proposed in [12] to efficiently compute the solutions corresponding to different values of the regularization parameter $\tau_1 > \tau_2 > \cdots > \tau_T$, often called *regularization path*. The idea is that the solution corresponding to the larger value τ_1 can be usually computed in a fast way since it is very sparse. Then, for τ_k one proceeds using the previously computed solution as the starting point of the corresponding procedure. It can be observed that with this *warm starting* much fewer iterations are typically required to achieve convergence.

4 Examples

In this section we illustrate the specialization of the framework described in the previous sections to a number structured sparsity regularization schemes.

4.1 Lasso and Elastic Net Regularization

We start considering the following functional

$$\mathcal{E}^{(\ell_1 \ell_2)}_{\tau,\mu}(\beta) = \|\Psi\beta - y\|^2 + \mu \sum_{j=1}^{d} \beta_j^2 + 2\tau \sum_{j=1}^{d} w_j |\beta_j|, \tag{9}$$

where Ψ is a $n \times d$ matrix, β, y are the vectors of coefficients and measurements respectively, and $(w_j)_{j=1}^d$ are positive weights. The matrix Ψ is given by the features ψ_j in the dictionary evaluated at some points $x_1, \ldots, x_n$.

Minimization of the above functional corresponds to the so called elastic net regularization, or ℓ_1-ℓ_2 regularization, proposed in [27], and reduces to the lasso algorithm [22] if we set $\mu = 0$. Using the notation introduced in the previous sections, we set $F(\beta) = \|\Psi\beta - y\|^2$ and $J(\beta) = \sum_{j=1}^d w_j |\beta_j|$. Moreover we denote by $\mathbf{S}_{\tau/\sigma}$ the soft-thresholding operator defined component-wise as in (7). The minimizer of (9) can be computed via the iterative update in Algorithm 2.

Note that the iteration in Algorithm 2 with $\mu = 0$ leads to the iterated soft-thresholding studied in [7] (see also [24] and references therein). When $\mu > 0$, the same iteration becomes the damped iterated soft-thresholding proposed in [8]. In the former case, the operator $\mathcal{T}_\sigma$ introduced in Theorem 1 is not contractive but only non-expansive, nonetheless convergence is still ensured [7].

Algorithm 2. Iterative Soft thresholding

Require: $\tau, \sigma > 0$
Initialize: $\beta^0 = 0$
while `convergence not reached` **do**
$p := p + 1$

$$\beta^p = \mathbf{S}_{\frac{\tau}{\sigma}} \left((1 - \frac{\mu}{\sigma})\beta^{p-1} + \frac{1}{\sigma}\Psi^T(y - \Psi\beta^{p-1}) \right)$$

end while
return β^p

Algorithm 3. Group lasso Algorithm

Require: $\tau, \sigma > 0$
Initialize: $\beta^0 = 0$
while `convergence not reached` **do**
$p := p + 1$

$$\beta^p = \tilde{\mathbf{S}}_{\frac{\tau}{\sigma}} \left((1 - \frac{\mu}{\sigma})\beta^{p-1} + \frac{1}{\sigma}\Psi^T(y - \Psi\beta^{p-1}) \right)$$

end while
return β^p

4.2 Group Lasso

We consider a variation of the above algorithms where the features are assumed to be composed in *blocks*. This latter assumption is used in [25] to define the so called group lasso, which amounts to minimizing

$$\mathcal{E}^{(grLasso)}_{\tau,\mu}(\beta) = \|\Psi\beta - y\|^2 + \mu \|\beta\|^2 + 2\tau \sum_{k=1}^{M} w_k \sqrt{\sum_{j \in \mathcal{I}_k} \beta_j^2} \tag{10}$$

for $\mu = 0$, where $(\psi_j)_{j \in \mathcal{I}_k}$ for $k = 1, \ldots, M$ is a block partition of the feature set $(\psi_j)_{j \in \mathcal{I}}$. If we define $\beta^{(k)} \in \mathbb{R}^{|\mathcal{I}_k|}$ the vector built with the components of $\beta \in \mathbb{R}^{|\mathcal{I}|}$ corresponding to the elements $(\psi_j)_{j \in \mathcal{I}_k}$, then the nonlinear operation $(I - \pi_{\lambda K})$ – denoted by $\tilde{\mathbf{S}}_{\tau/\sigma}$ – acts on each block as (7), and the minimizer of (10) can hence be computed through Algorithm 3.

4.3 Composite Absolute Penalties

In [26], the authors propose a novel penalty, named Composite Absolute Penalty (CAP), based on assuming possibly overlapping groups of features. Given $\gamma_k \in \mathbb{R}^+$, for $k = 0, 1, \ldots, M$, the penalty is defined as:

$$J(\beta) = \sum_{k=1}^{M} (\sum_{j \in \mathcal{I}_k} \beta_j^{\gamma_k})^{\frac{\gamma_0}{\gamma_k}},$$

where $(\psi_j)_{j \in \mathcal{I}_k}$ for $k = 1, \ldots, M$ is not necessarily a block partition of the feature set $(\psi_j)_{j \in \mathcal{I}}$. This formulation allows to incorporate in the model not only groupings, but also hierarchical structures present within the features, for

Algorithm 4. CAP Algorithm

Require: $\tau, \sigma > 0$
Initialize: $\beta^0 = 0, p = 0, \bar{v}^0 = 0$
while `convergence not reached` **do**
$p := p + 1$

$$\tilde{\beta} = (1 - \frac{\mu}{\sigma})\beta^{p-1} + \frac{1}{\sigma}\Psi^T(y - \Psi\beta^{p-1})$$

set $v^0 = \bar{v}_{p-1}$
while `convergence not reached` **do**
$q := q + 1$
for k=1,...,M **do**

$$v_k^q = (\pi_{\frac{\tau}{\eta}B})_k \left(v_k^{q-1} - \frac{1}{\eta}\mathcal{J}_k(\mathcal{J}^T v^{q-1} - \tilde{\beta}) \right)$$

end for
$\bar{v} = v^q$
end while

$$\beta^p = \tilde{\beta} - \frac{\tau}{\sigma}\mathcal{J}^T\bar{v}_p$$

end while
return β^p

instance by setting $\mathcal{I}_k \subset \mathcal{I}_{k-1}$. For $\gamma_0 = 1$, the CAP penalty is one-homogeneous and the solution can be computed through Algorithm 1. Furthermore, when $\gamma_k = 2$ for all $k = 1, \ldots, M$, it can be regarded as a particular case of (3), with $\|\mathcal{J}_k(\beta)\|^2 = \sum_{j=1}^d \beta_j^2 \mathbf{1}_{\mathcal{I}_k}(j)$, with $\mathcal{J}_k : \mathbb{R}^{|\mathcal{I}|} \to \mathbb{R}^{|\mathcal{I}_k|}$. Considering the least square error, we study the minimization of the functional

$$\mathcal{E}_{\tau,\mu}^{(CAP)}(\beta) = \|\Psi\beta - y\|^2 + \mu\|\beta\|^2 + 2\tau\sum_{k=1}^M w_k \sqrt{\sum_{j\in\mathcal{I}_k} \beta_j^2}, \tag{11}$$

which is a CAP functional when $\mu = 0$. Note that, due to the overlapping structure of the features groups, the minimizer of (11) cannot be computed blockwise as in Algorithm 3, and therefore need to combine it with the iterative update (8) for the projection, thus obtaining Algorithm 4.

4.4 Multiple Kernel Learning

Multiple kernel learning (MKL) [2, 18] is the process of finding an optimal kernel from a prescribed (convex) set $\mathcal{K}$ of basis kernels, for learning a real-valued function by regularization. In the following we consider the case where the set $\mathcal{K}$ is the convex hull of a finite number of kernels $k_1, \ldots, k_M$, and the loss function is the square loss. It is possible to show [17] that the problem of multiple kernel learning corresponds to find f^* solving

$$\underset{f\in\mathcal{H}}{\operatorname{argmin}} \left\{ \frac{1}{n}\sum_{i=1}^n (\sum_{j=1}^M f_j(x_i) - y_i)^2 + \mu\sum_{j=1}^M \|f_j\|_{\mathcal{H}_j}^2 + 2\tau\sum_{j=1}^M \|f_j\|_{\mathcal{H}_j} \right\}, \tag{12}$$

for $\mu = 0$, with $\mathcal{H} = \mathcal{H}_{k_1} \oplus \cdots \oplus \mathcal{H}_{k_M}$.

Algorithm 5. MKL Algorithm

set $\alpha^0 = 0$
while `convergence not reached` **do**
$p := p + 1$

$$\alpha^p = \hat{\mathbf{S}}_{\tau/\sigma}\left(K, \left((1 - \frac{\mu}{\sigma})\alpha^{p-1} - \frac{1}{\sigma n}(\mathrm{K}\alpha^{p-1} - \mathbf{y})\right)\right)$$

end while
return $(\alpha^p)^T \mathbf{k}$.

Note that our general hypotheses on the penalty term J are clearly satisfied. Though the space of functions is infinite dimensional, thanks to a generalization of the representer theorem, the minimizer of the above functional (12) can be shown to have the finite representation $f_j^*(\cdot) = \sum_{i=1}^n \alpha_{j,i} k_j(x_i, \cdot)$ for all $j = 1, \ldots, M$. Furthermore, introducing the following notation:

$$\alpha = (\alpha_1, \ldots, \alpha_M)^T \text{ with } \alpha_j = (\alpha_{j,1}, \ldots, \alpha_{j,n})^T,$$
$$\mathbf{k}(x) = (\mathbf{k}_1(x), \ldots, \mathbf{k}_M(x))^T \qquad \text{with } \mathbf{k}_j(x) = (k_j(x_1, x), \ldots, k_j(x_n, x)),$$
$$\mathrm{K} = \left.\begin{pmatrix} \mathrm{K}_1 \ldots \mathrm{K}_M \\ \vdots \ \ddots \ \vdots \\ \mathrm{K}_1 \ldots \mathrm{K}_M \end{pmatrix}\right\} M \text{ times} \qquad \text{with } [\mathrm{K}_j]_{ii'} = k_j(x_i, x_i'),$$
$$\mathbf{y} = (\underbrace{y^T, \ldots, y^T}_{M \text{ times}})^T$$

we can write the solution of (12) as $f^*(x) = \alpha_1^T \mathbf{k}_1(x) + \cdots + \alpha_M^T \mathbf{k}_M(x)$, which coefficients can be computed using Algorithm 5, where the soft-thresholding operator $\hat{\mathbf{S}}_\lambda(\mathrm{K}, \alpha)$ acts on the components α_j as

$$\hat{\mathbf{S}}_\lambda(\mathrm{K}, \alpha)_j = \frac{\alpha_j^T}{\sqrt{\alpha_j^T \mathrm{K}_j \alpha_j}}(\sqrt{\alpha_j^T \mathrm{K}_j \alpha_j} - \lambda)_+.$$

4.5 Multitask Learning

Learning multiple tasks simultaneously has been shown to improve performance relative to learning each task independently, when the tasks are related in the sense that they all share a small set of features (see for example [1, 14, 20] and references therein). In particular, given T tasks modeled as $f_t(x) = \sum_{j=1}^d \beta_{j,t} \psi_j(x)$ for $t = 1, \ldots, T$, according to [20], regularized multi-task learning amounts to the minimization of the functional

$$\mathcal{E}_{\tau,\mu}^{(MT)}(\beta) = \sum_{t=1}^T \frac{1}{n_t} \sum_{i=1}^{n_t} (\psi(x_{t,i})\beta_t - y_{t,i})^2 + \mu \sum_{t=1}^T \sum_{j=1}^d \beta_{t,j}^2 + 2\tau \sum_{j=1}^d \sqrt{\sum_{t=1}^T \beta_{t,j}^2}. \tag{13}$$

Algorithm 6. Multi-Task Learning Algorithm

set $\beta^0 = 0$
while `convergence not reached` **do**
$p := p + 1$

$$\beta^p = \tilde{\mathbf{S}}_{\frac{\tau}{\sigma}} \left((1 - \frac{\mu}{\sigma})\beta^{p-1} + \frac{1}{\sigma}\Psi^T \mathbf{N}(y - \Psi\beta^{p-1}) \right)$$

end while
return β^p

The last term combines the tasks and ensures that common features will be selected across them. Functional (13) is a particular case of (1), and, defining

$$\begin{aligned}
&\beta = (\beta_1^T, \ldots, \beta_T^T)^T, \\
&\Psi = \mathrm{diag}(\Psi_1, \ldots, \Psi_T), \qquad [\Psi_t]_{ij} = \psi_j(x_{t,i}), \\
&y = (y_1^T, \ldots, y_T^T)^T, \\
&\mathbf{N} = \mathrm{diag}(\underbrace{1/n_1, \ldots, 1/n_1}_{n_1 \text{times}}, \underbrace{1/n_2, \ldots, 1/n_2}_{n_2 \text{times}}, \ldots, \underbrace{1/n_T, \ldots, 1/n_T}_{n_T \text{times}}).
\end{aligned}$$

its minimizer can be computed through Algorithm 6. The Soft-thresholding operator $\tilde{\mathbf{S}}_\lambda$ is applied task-wise, that is it acts simultaneously on the regression coefficients relative to the same variable in all the tasks.

5 Experiments and Discussions

In this section we describe several experiments aimed at testing some features of the proposed method. In particular, we investigate the effect of adding the term $\mu \|f\|^2_{\mathcal{H}}$ to the original functional in terms of
-*prediction*: do different values of μ modify the prediction error of the estimator?
-*selection*: does μ increase/decrease the sparsity level of the estimator?
-*running time*: is there a computational improvement due to the use of $\mu > 0$?

We discuss the above questions for the multi-task scheme proposed in [20]. and show results which are consistent with those reported in [9] for the elastic-net estimator. These two methods are only two special cases of our framework, but indeed we expect that, due to the common structure of the penalty terms, all the other learning algorithms considered in this paper share the same properties.

We note that a computational comparison of different optimization approaches is cumbersome since we consider many different learning schemes and is beyond the scope of this paper. Extensive analysis of different approaches to solve ℓ_1 regularization can be found in [12] and [15], where the authors show that projected gradient methods compare favorably to state of the art methods. We expect that similar results will hold for learning schemes other than ℓ_1 regularization.

5.1 Validation Protocol and Simulated Data

In this section, we briefly present the set-up used in the experiments. We considered simulated data to test the properties of the proposed method in a controlled scenario. More precisely, we considered T regression tasks

$$y = x \cdot \beta_t + \varepsilon \quad t = 1, \ldots, T$$

where x is uniformly drawn from $[0,1]^d$, ε is drawn from the zero-mean Gaussian distribution with $\sigma = 0.1$ and the regression vectors are

$$\beta_t^\dagger = (\beta_{t,1}^\dagger, ..., \beta_{t,r}^\dagger, 0, 0, ..., 0).$$

with $\beta_{t,j}^\dagger$ uniformly drawn from $[-1,1]$ for $t \leq r$, so that the relevant variables are the first r.

Following [5, 23], we consider a debiasing step after running the sparsity based procedure. This last step is a post-processing and corresponds to training a regularized least square (RLS) estimator[1] with parameter λ on the selected components to avoid an undesired shrinkage of the corresponding coefficients.

In order to obtain a fully data driven procedure we use cross validation to choose the regularization parameters τ, λ. After re-training with the optimal regularization parameters, a test error is computed on an independent set of data. Each validation protocol is replicated 20 times by resampling both the input data and the regression coefficients, $\beta_t^\dagger$, in order to assess the stability of the results.

5.2 Role of the Strictly Convex Penalty

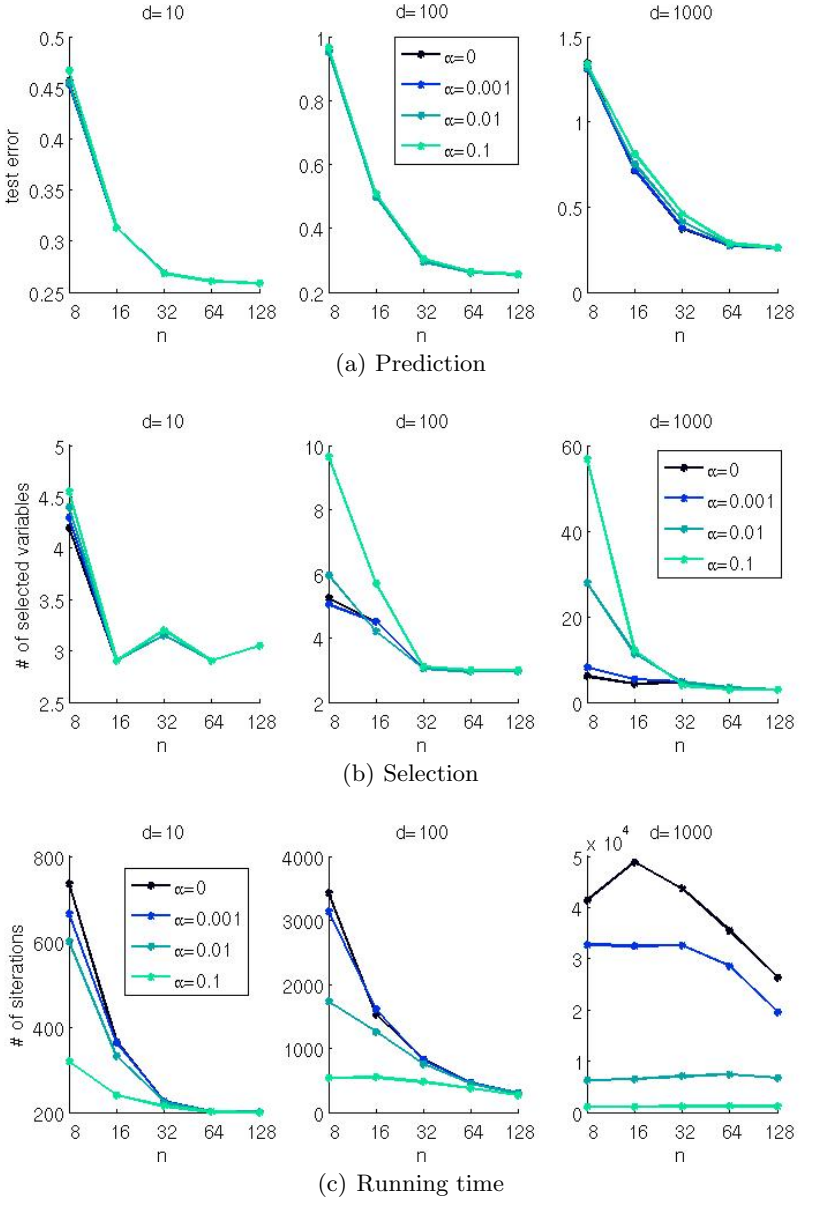
We investigate the impact of adding the perturbation $\mu > 0$. We consider $T = 2$, $r = 3$, $d = 10, 100, 1000$, and $n = 8, 16, 32, 64, 128$. For each data set, that is for fixed d and n, we apply the validation protocol described in Subsection 5.1 for increasing values of μ. The number of samples in the validation and test sets is 1000. Error bars are omitted in order to increase the readability of the Figures.

We preliminary discuss an observation suggesting a useful way to vary μ. As a consequence of (5), when $\mu = 0$ and $b = 0$, the Lipschitz constant, L_σ, of the map $\mathcal{T}_\sigma$ in Theorem 1 is 1 so that $\mathcal{T}_\sigma$ is not a contraction. By choosing $\mu = \frac{1}{4}\left\|\nabla^2 F\right\| \alpha$ with $\alpha > 0$, the Lipschitz constant becomes $L_\sigma = (1+\alpha)^{-1} < 1$, and the map $\mathcal{T}_\sigma$ induced by F_μ is a contraction. In particular in multiple task learning with linear features (see Section 4.5) $X = \Psi$, so that $\nabla^2 F = 2X^T X/n$ and $\left\|\nabla^2 F\right\| = 2a/n$, where a is the largest eigenvalue of the symmetric matrix $X^T X$. We therefore let $\mu = \frac{a}{2n}\alpha$ and vary the absolute parameter α as $\alpha = 0,\ 0.001,\ 0.01,\ 0.1$. We then compare the results obtained for different values of α, and analyze in the details the outcome of our results in terms of the three aspects raised at the beginning of this section.

-*prediction* The test errors associated to different values of μ are essentially overlapping, meaning that the perturbation term does not impact the prediction performance of the algorithm when the τ parameter is accurately tuned. This result is consistent with the theoretical results for elastic net – see [8].

-*selection* In principle the presence of the perturbation term tends to reduce the sparsity of the solution in the presence of very small samples. In practice

[1] A simple ordinary least square is often sufficient and here a little regularization is used to avoid possible unstable behaviors especially in the presence of small samples.



(a) Prediction

(b) Selection

(c) Running time

Fig. 1. Results obtained in the experiments varying the size of the training set and the number of input variables. The properties of the algorithms are evaluated in terms of the prediction error, the ability of selecting the *true* relevant variables, and finally the number of iteration required for the convergence of the algorithm.

one can see that such an effect decreases when the number of input points n increases and is essentially negligible even when $n << d$.

- *running time* From the computational point of view we expect larger values of μ (or equivalently α) to correspond to fewer iterations. This effect is clear in our experiments. Interestingly when $n << d$ small values of μ allow to substantially reduce the computational burden while preserving the prediction property of the algorithm (compare α=0 and α=0.001 for d=1000). Moreover, one can observe that the number of iterations decreases as the number of points increases. This result might seem surprising, but can be explained recalling that the condition number of the underlying problem is likely to improve as n increases.

Finally, we can see that adding the small strictly convex perturbation with $\mu > 0$, has a preconditioning effect on the iterative procedure and can substantially reduce the number of required computations without affecting the prediction property of the obtained solution.

5.3 Impact of Choosing the Step-Size Adaptively

In this section we assess the effectiveness of the adaptive approach proposed in section 3.3 to speed up the convergence of the algorithm. Specifically, we show some results obtained by running the iterative optimization with two different choices of the step-size, namely the one fixed a-priori – as described in section 3.1 – and the adaptive alternative of subsection 3.3.

The experiments have been conducted by first drawing randomly the dataset and finding the optimal solution using the complete validation scheme, and then running two further experiments using, in both cases, the optimal regularization parameters but the two different strategies for the step-size.

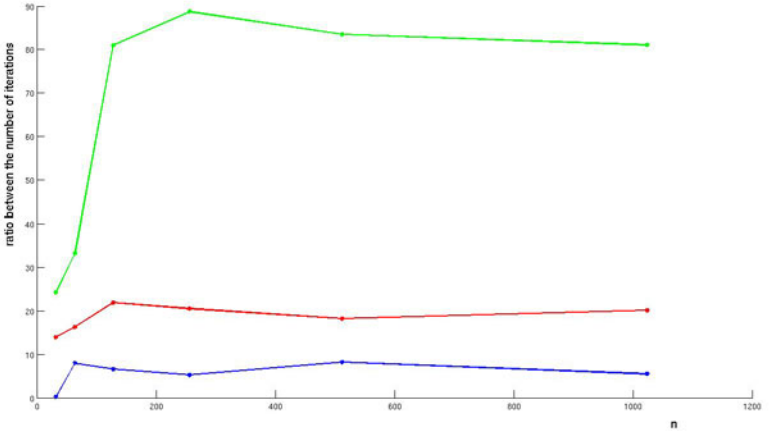


Fig. 2. Comparison of the number of iterations required to compute the regression function using the fixed and the adaptive step-size. The blue plot refers to the experiments using $d = 10$, the red plot to $d = 100$, while the green plot to $d = 500$.

We compared the number of iterations necessary to compute the solution and looked at the ratio between those required by the fixed and the adaptive strategies respectively. In Figure 2, it is easy to note that such ratio is always greater than one, and actually it ranges from the order of tens to the order of hundreds. Moreover, the effectiveness of using an adaptive strategy becomes more and more evident as the number of input variables increases. Finally, for a fixed input dimension, the number of iterations required for both choices of the step-size decreases when the number of training samples increases, in a way that the ratio tends to either remain approximately constant or decrease slightly.

6 Conclusions

This paper shows that many algorithms based on regularization with convex non differentiable penalties can be described within a common framework. This allows to derive a general optimization procedure based on proximal methods whose convergence is guaranteed. The proposed procedure highlights and separates the roles played by the loss terms and the penalty terms, in fact, it corresponds to the iterative projection of the gradient of the loss on a set defined by the penalty. The projection has a simple characterization in the setting we consider: in many cases it can be written in closed form and corresponds to a soft-thresholding operator, in all the other cases it can be iteratively calculated by resorting again to proximal methods. The obtained procedure is simple and its convergence proof is straightforward in the strictly convex case. One can always force such a condition considering a suitable perturbation of the original functional. Interestingly if such a perturbation is small it will act as a preconditioning of the problem and lead to the better computational performances without changing the properties of the solution.

Acknowledgments

This work has been partially supported by the FIRB project LEAP RBIN04PARL, the EU Integrated Project Health-e-Child IST-2004-027749. Matteo Santoro and Sofia Mosci are partially supported by Compagnia di San Paolo. This report describes research done at the Center for Biological & Computational Learning, which is in the McGovern Institute for Brain Research at MIT, as well as in the Dept. of Brain & Cognitive Sciences, and which is affiliated with the Computer Sciences & Artificial Intelligence Laboratory (CSAIL). This research was sponsored by grants from DARPA (IPTO and DSO), National Science Foundation (NSF-0640097, NSF-0827427.) Additional support was provided by: Adobe, Honda Research Institute USA, King Abdullah University Science and Technology grant to B. DeVore, NEC, Sony and especially by the Eugene McDermott Foundation.

References

[1] Argyriou, A., Hauser, R., Micchelli, C.A., Pontil, M.: A dc-programming algorithm for kernel selection. In: Proceedings of the Twenty-Third International Conference on Machine Learning (2006)

[2] Bach, F.R., Lanckriet, G., Jordan, M.I.: Multiple kernel learning, conic duality, and the smo algorithm. In: ICML. ACM International Conference Proceeding Series, vol. 69 (2004)

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

[4] Becker, S., Bobin, J., Candes, E.: Nesta: A fast and accurate first-order method for sparse recovery (2009)

[5] Candès, E., Tao, T.: The Dantzig selector: statistical estimation when p is much larger than n. Ann. Statist. 35(6), 2313–2351 (2005)

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

[7] Daubechies, I., Defrise, M., De Mol, C.: An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Communications on Pure and Applied Mathematics 57, 1413–1457 (2004)

[8] De Mol, C., De Vito, E., Rosasco, L.: Elastic-net regularization in learning theory (2009)

[9] De Mol, C., Mosci, S., Traskine, M., Verri, A.: A regularized method for selecting nested groups of relevant genes from microarray data. Journal of Computational Biology, 16 (2009)

[10] Dontchev, A.L., Zolezzi, T.: Well-posed optimization problems. Lecture Notes in Mathematics, vol. 1543. Springer, Heidelberg (1993)

[11] Duchi, J., Singer, Y.: Efficient online and batch learning using forward backward splitting. Journal of Machine Learning Research 10, 2899–2934 (2009)

[12] Hale, E.T., Yin, W., Zhang, Y.: Fixed-point continuation for l1-minimization: Methodology and convergence. SIOPT 19(3), 1107–1130 (2008)

[13] Jenatton, R., Audibert, J.-Y., Bach, F.: Structured variable selection with sparsity-inducing norms. Technical report, INRIA (2009)

[14] Kubota, R.A., Zhang, T.: A framework for learning predictive structures from multiple tasks and unlabeled data. J. Mach. Learn. Res. 6, 1817–1853 (2005)

[15] Loris, I.: On the performance of algorithms for the minimization of l_1-penalized functionals. Inverse Problems 25(3) 035008, 16 (2009)

[16] Loris, I., Bertero, M., De Mol, C., Zanella, R., Zanni, L.: Accelerating gradient projection methods for ℓ_1-constrained signal recovery by steplength selection rules (2009)

[17] Micchelli, C.A., Pontil, M.: Learning the kernel function via regularization. J. Mach. Learn. Res. 6, 1099–1125 (2005)

[18] Micchelli, C.A., Pontil, M.: Feature space perspectives for learning the kernel. Mach. Learn. 66(2-3), 297–319 (2007)

[19] Nesterov, Y.: Smooth minimization of non-smooth functions. Math. Program. 103(1), 127–152 (2005)

[20] Obozinski, G., Taskar, B., Jordan, M.I.: Multi-task feature selection. Technical report, Dept. of Statistics, UC Berkeley (June 2006)

[21] Rosasco, L., Mosci, S., Santoro, A., Verri, M., Villa, S.: Iterative projection methods for structured sparsity regularization. Technical Report MIT-CSAIL-TR-2009-050 CBCL-282 (October 2009)

[22] Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, Series B 56, 267–288 (1996)
[23] Wright, S.J., Nowak, R.D., Figueiredo, M.A.T.: Sparse reconstruction by separable approximation. IEEE Trans. Image Process (2009)
[24] Yin, W., Osher, S., Goldfarb, D., Darbon, J.: Bregman iterative algorithms for ℓ^1-minimization with applications to compressed sensing. SIAM J. Imaging Sciences 1(1), 143–168 (2008)
[25] Yuan, M., Lin, Y.: Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society, Series B 68(1), 49–67 (2006)
[26] Zhao, P., Rocha, G., Yu, B.: The composite absolute penalties family for grouped and hierarchical variable selection. Annals of Statistics 37(6A), 3468–3497 (2009)
[27] Zou, Z., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society, Series B 67, 301–320 (2005)