ν-SVM Solutions of Constrained Lasso and Elastic Net

Alberto Torres-Barrán[a,b], Carlos M. Alaíz[a,*], José R. Dorronsoro[a,c]

[a]*Dpto. Ing. Informática, Universidad Autónoma de Madrid, 28049 Madrid, Spain*
[b]*Inst. de Ciencias Matemáticas ICMAT, Campus Cantoblanco UAM, 28049 Madrid, Spain*
[c]*Inst. de Ing. del Conocimiento, Campus Cantoblanco UAM, 28049 Madrid, Spain*

Abstract

Many important linear sparse models have at its core the Lasso problem, for which the GLMNet algorithm is often considered as the current state of the art. Recently M. Jaggi has observed that Constrained Lasso (CL) can be reduced to an SVM-like problem, for which the LIBSVM library provides very efficient algorithms. This suggests that it could also be used advantageously to solve CL. In this work we will refine Jaggi's arguments to reduce CL as well as constrained Elastic Net to a Nearest Point Problem, which in turn can be rewritten as an appropriate ν-SVM problem solvable by LIBSVM. We will also show experimentally that the well-known LIBSVM library results in a faster convergence than GLMNet for small problems and also, if properly adapted, for larger ones. Screening is another ingredient to speed up solving Lasso. Shrinking can be seen as the simpler alternative of SVM to screening and we will discuss how it also may in some cases reduce the cost of an SVM-based CL solution.

Keywords: Lasso, GLMNet, Nearest Point Problem, SVM

1. Introduction

The pervasiveness of Big Data is putting a strong emphasis in simple linear sparse models as a way to handle large size and large dimensional samples. This has led to position Lasso [1] and its Elastic Net generalization [2] among the

*Corresponding author.
Email addresses: `alberto.torres@uam.es` (Alberto Torres-Barrán), `carlos.alaiz@inv.uam.es` (Carlos M. Alaíz), `jose.dorronsoro@uam.es` (José R. Dorronsoro)

methods of choice in large-scale Machine Learning. The more general Elastic Net is usually stated as the unconstrained problem of solving

$$\min_{\beta \in \mathbb{R}^d} \left\{ \frac{1}{2} \|X\beta - y\|_2^2 + \frac{\mu}{2} \|\beta\|_2^2 + \lambda \|\beta\|_1 \right\} , \tag{1}$$

where for an N-size sample $S = \left\{(x^1, y^1), \ldots, (x^N, y^N)\right\}$, X is an $N \times d$ data matrix containing as its rows the transposes of the d-dimensional features x^n, y is the $N \times 1$ target vector $y = \left(y^1, \ldots, y^N\right)^\top$, β denotes the Elastic Net coefficients, and the subscripts 1, 2 denote the ℓ_1 and ℓ_2 norms respectively. We will call problem (1) (λ, μ)-Unconstrained Elastic Net (UEN). It has an equivalent constrained formulation:

$$\min_{\beta \in \mathbb{R}^d} \left\{ \frac{1}{2} \|X\beta - y\|_2^2 + \frac{\mu}{2} \|\beta\|_2^2 \right\} \quad \text{s.t. } \|\beta\|_1 \leq \rho . \tag{2}$$

As before, we will call (2) (ρ, μ)-Constrained Elastic Net (CEN). Problems (1) and (2) obviously reduce to Lasso when $\mu = 0$ and in a slight abuse of language, we will also refer to them simply as λ-Unconstrained Lasso (UL) or ρ-Constrained Lasso (CL), respectively.

When solving (1), the GLMNet algorithm [3], which uses cyclic coordinate descent, stands out in terms of computational efficiency. On the other hand, (1) and other problems that decompose as a sum of convex differentiable and non-differentiable functions can be solved using the well-known FISTA algorithm [4], based on proximal theory and that combines projected gradient descent with Nesterov's acceleration.

Recently M. Jaggi [5] has shown the equivalence between CL and SVMs or, more precisely, a Nearest Point Problem (NPP), in the sense that given any instance of either problem, one can construct an instance of the other having the same optimal solution. This is relatively simple when going from Lasso to NPP but more involved the other way around. As suggested in [5], this equivalence opens the way for advances in one problem to be translated into advances on the other and while the application of SVM solvers to Lasso is not addressed in [5], prior work in parallelizing SVMs is leveraged in [6] to obtain a highly optimized and parallel solver for Elastic Net taking advantage of existing work on extending SVM solvers to GPUs, multi-core CPUs and distributed systems. Lasso is not addressed in [6] but some of its authors have considered the Lasso-only case in [7] relating it with a minimum norm problem that they solve using Wolfe's method.

In this work we will retake Jaggi's approach, and simplify the Lasso and Elastic Net reduction in [5] to yield a ν-SVM problem equivalent to initial (ρ, μ)-CEN or ρ-CL ones. We shall then apply the LIBSVM solver for ν-SVM and its underlying SMO algorithm, first in a straightforward way and then refining it so that it takes advantage of the particular form of the kernel matrix of the resulting ν-SVM problem. Furthermore, we shall consider shrinking as the SMO alternative to the recently proposed screening methods [8, 9] for Lasso, and discuss the reduction that it provides in training times. More precisely, our contributions are:

- A proof of the explicit equivalence between unconstrained and constrained Lasso. While a general equivalence can be derived using primal and dual considerations in convex optimization (see Sect. 2 for details), these cannot be used to establish explicitly the relationship between the constrained constant ρ and the unconstrained one λ. Here we give this specific relation between the regularization parameters of the unconstrained problem and the equivalent constrained one.

- A refinement on M. Jaggi's results reducing CL to NPP and its further reduction to a ν-SVM problem solvable using LIBSVM.

- A comparison of number of iterations and execution times of the GLMNet algorithm for UL and of two SVM-based solutions of CL, a first one where LIBSVM is used with no changes, and a second one where we modify the handling of the kernel matrix of LIBSVM to take advantage of its particular structure on the Lasso-to-NPP reduction. As we shall see, both approaches are competitive with GLMNet, currently the state-of-the-art algorithm for Lasso, with the second one being faster on most of the problems considered.

The paper is organized as follows. In Sect. 2 we first recall the easy reduction of UEN and CEN to UL and CL over an extended sample, prove then the equivalence between UL and CL and extend it to that between UEN and CEN. In Sect. 3 we discuss the reduction of CL and CEN to NPP and ν-SVMs and we briefly describe in Sect. 4 the FISTA, GLMNet and SMO algorithms and their complexities. In Sect. 5 we give a relatively wide review of various screening techniques that have been proposed to reduce the working set of Lasso (i.e., the subset of active features) and we discuss the shrinking technique of SVM as a counterpart to the screening of Lasso which may yield further computational advantages. Section 6 contains experimental comparisons of the performances of all the methods considered, and the paper closes with a brief discussion and pointers to further work in Sect. 7.

2. Unconstrained and Constrained Lasso and Elastic Net

In this section we show that λ-CL and ρ-UL and also (μ, λ)-UEN and (μ, ρ)-CEN are equivalent problems in the sense that they have the same β solution for appropriate choices of λ and ρ. We will first reduce Elastic Net (EN) to a pure Lasso problem over an enlarged data matrix and target vector; the basic idea has been already used quite often, for instance, when reducing soft margin, square hinge loss SVMs to hard margin SVMs. To do so, let's consider in either problem (1) or (2) the $(N+d) \times d$ matrix $\mathcal{X}$ and $(N+d) \times 1$ vector $\mathcal{Y}$ defined as

$$\mathcal{X} = \begin{pmatrix} X \\ \sqrt{\mu} I_d \end{pmatrix} , \quad \mathcal{Y} = \begin{pmatrix} y \\ \mathbf{0} \end{pmatrix} ,$$

with I_d the $d \times d$ identity matrix and $\mathbf{0}$ the d-dimensional 0 vector. We then have

$$\frac{1}{2} \|X\beta - y\|_2^2 + \frac{\mu}{2} \|\beta\|_2^2 = \frac{1}{2} \|\mathcal{X}\beta - \mathcal{Y}\|_2^2 ,$$

where $\mathcal{X}$ and $\mathcal{Y}$ are the new data matrix and target vector. In other words, we can rewrite (1) or (2) as Lasso problems over an extended sample of size $N + d$, namely:

$$\mathcal{S} = \left((x^1, y^1), \dots, (x^N, y^N), (\sqrt{\mu}e_1, 0), \dots, (\sqrt{\mu}e_d, 0)\right) ,$$

with e_i the canonical vectors for $\mathbb{R}^d$. As a consequence, we limit from now on our discussion to UL and CL, pointing when necessary to the changes needed to deal with UEN and CEN. For simplicity we will revert to the X and y notation instead of using $\mathcal{X}$ and $\mathcal{Y}$ when referring to the data matrix and target vector of Lasso.

Turning our attention to the UL and CL equivalence, it is clear that, for a fixed λ, a minimizer $\beta(\lambda)$ of λ-UL is also a minimizer of ρ-CL for $\rho = \|\beta(\lambda)\|_1$, as any better minimizer of (2) would also automatically be a better minimizer of (1). On the other hand, given a general constrained (primal) convex problem of the form

$$\min_{\beta} \ \{f(\beta)\} \quad \text{s.t.} \ g(\beta) - \rho \leq 0 ,$$

it is easy to see that its solution β^* also solves its unconstrained version

$$\min_{\beta} \ \{f(\beta) + \lambda_\rho g(\beta)\}$$

for some λ_ρ which depends on ρ. In fact, the Lagrangian of the unconstrained problem is $L(\beta, \lambda) = f(\beta) + \lambda(g(\beta) - \rho)$, $\lambda \geq 0$, and if β^* is its primal solution and λ^* is its dual one, under rather general conditions it follows [10, Subsection 5.2.3] that the dual gap is 0 and thus

$$L(\beta^*, \lambda^*) = f(\beta^*) + \lambda^*(g(\beta^*) - \rho) = f(\beta^*) = \min_{\{\beta : g(\beta) \leq \rho\}} \ \{f(\beta)\} .$$

Now, since the term $-\rho\lambda^*$ is independent of β, it thus follows that β^* also minimizes the unconstrained problem for $\lambda^* = \lambda_\rho$, i.e.

$$\min_{\beta} \ \{f(\beta) + \lambda^* g(\beta)\} .$$

In particular, all this holds to the Lasso problem but, however, in this general setting there is no way of explicitly deriving from β^* and ρ the concrete λ_ρ value needed to write down the general unconstrained problem.

In order to do so for Lasso, let β_{LR} be the minimum norm solution of Linear Regression and set $\rho_{\mathrm{LR}} = \|\beta_{\mathrm{LR}}\|_1$. Clearly the solution of CL is also β_{LR} for $\rho \geq \rho_{\mathrm{LR}}$, so we will assume $\rho < \rho_{\mathrm{LR}}$, and hence the constraint is active. Let's denote by β_ρ a minimum of ρ-CL; we shall prove next that β_ρ solves λ_ρ-UL with

$$\lambda_\rho = \frac{\beta_\rho^\top X^\top (y - X\beta_\rho)}{\rho} = \frac{\beta_\rho^\top X^\top r^{\beta_\rho}}{\rho} \ , \tag{3}$$

with r^{β_ρ} denoting the residual $y - X\beta_\rho$. To do so, let $e_2(\beta) = \|X\beta - y\|_2^2/2$ and $g(\beta) = \|\beta\|_1 - \rho$, let $e_2^\rho = e_2(\beta_\rho)$ be the optimal square error in ρ-CL, and define

$$f(\beta) = \max\{e_2(\beta) - e_2^\rho, g(\beta)\} \ .$$

Note that the convex function f verifies $f(\beta) \geq 0$. In fact, if $g(\beta) \leq 0$,

$$f(\beta) \geq e_2(\beta) - e_2^\rho \geq 0 \ ,$$

since otherwise β_ρ would not be a solution of ρ-CL. On the other side, for $g(\beta) > 0$,

$$f(\beta) \geq g(\beta) > 0 \ .$$

Now, since $f(\beta_\rho) = 0$, it follows that β_ρ minimizes f and as a consequence, $0 \in \partial f(\beta_\rho)$. Moreover, the subgradient $\partial f(\beta_\rho)$ is given by the convex hull generated by $\partial e_2(\beta_\rho) = \{\nabla e_2(\beta_\rho)\}$ and $\partial g(\beta_\rho)$, that is,

$$\partial f(\beta_\rho) = \{\gamma \nabla e_2(\beta_\rho) + (1-\gamma)h : 0 \leq \gamma \leq 1, h \in \partial g(\beta_\rho)\} \ .$$

Thus, for some $h_\rho \in \partial \|\cdot\|_1 (\beta_\rho)$ and γ, with $0 \leq \gamma \leq 1$,

$$0 = \gamma \nabla e_2(\beta_\rho) + (1-\gamma) h_\rho \ .$$

But $\gamma > 0$, otherwise we would have $h = 0$, i.e., $\beta_\rho = 0$, contradicting that $\|\beta_\rho\|_1 = \rho$ since the constraint is active. Therefore, writing $\lambda_\rho = (1-\gamma)/\gamma$, we arrive at

$$0 = \nabla e_2(\beta_\rho) + \lambda_\rho h_\rho \in \partial[e_2(\cdot) + \lambda_\rho \|\cdot\|_1](\beta_\rho) \ ,$$

i.e., β_ρ minimizes λ_ρ-UL. We finally derive the value for λ_ρ in (3) by observing that $\beta_\rho^\top h = \|\beta_\rho\|_1 = \rho$ and that

$$0 = \nabla e_2(\beta_\rho) + \lambda_\rho h_\rho = -X^\top r^{\beta_\rho} + h_\rho \lambda_\rho$$

implies $h\lambda_\rho = X^\top r^{\beta_\rho}$. As a result,

$$\lambda_\rho \|\beta_\rho\|_1 = \beta_\rho^\top h \lambda_\rho = \beta_\rho^\top X^\top r^{\beta_\rho} \ ,$$

that is,

$$\lambda_\rho = \frac{\beta_\rho^\top X^\top r^{\beta_\rho}}{\|\beta_\rho\|_1} = \frac{\beta_\rho^\top X^\top r^{\beta_\rho}}{\rho} \ .$$

The corresponding λ_ρ for (μ, ρ)-CEN would be

$$\lambda_\rho = \frac{(\mathcal{X}\beta_\rho) \cdot (\mathcal{Y} - \mathcal{X}\beta_\rho)}{\rho} = \frac{\beta_\rho^\top X^\top (y - X\beta_\rho) - \mu \|\beta_\rho\|_2^2}{\rho} \ .$$

In summary, we have shown that λ-UL and ρ-CL are equivalent problems but, before we finish, we point out that they are so in the sense that a solution of one of them is also the solution of the other for an appropriate choice of either the λ or ρ parameters. However, the optimal ρ can only be found **after λ-UL has been solved** and vice-versa. Thus λ-UL and ρ-CL require each its own algorithm; in particular, algorithms for, say, λ-UL such as FISTA or GLMNet cannot be used to solve a given ρ-CL problem, since the equivalent λ_ρ value will only be known **after** the starting ρ-CL problem has already been solved.

3. From Constrained Lasso to NPP

This section reviews the connection between Lasso and NPP and discusses the references [6, 7], which are closely related to this work.

3.1. Connection between Lasso and NPP

As mentioned in the introduction, M. Jaggi proposes in [5] a reduction of ρ-CL to an SVM-like problem. Here we will initially follow [5], but making the reduction more precise, going first from ρ-CL to a particular case of the Nearest Point Problem (NPP) that we then rewrite as a ν-SVM problem.

We first rescale β and y by $1/\rho$ and work with $\tilde{\beta} = \beta/\rho$ and $\tilde{y} = y/\rho$; then ρ-CL on β and y becomes 1-CL on $\tilde{\beta}$ and $\tilde{y}$, with the new constraint being $\tilde{\beta} \in B_1$, the ℓ_1 unit ball. Now, since B_1 is convex, any $\beta \in B_1$ can be written as a convex combination of B_1's extreme points, i.e., the canonical vectors e_i and their negatives $-e_j$, $1 \leq i, j \leq d$. Thus, we can further replace B_1 by the simplex Δ_{2d} of $2d$ dimensional probability vectors, i.e.,

$$\Delta_{2d} = \left\{ \alpha \in \mathbb{R}^{2d} : 0 \leq \alpha_i \leq 1, \sum_{1}^{2d} \alpha_i = 1 \right\} ,$$

by enlarging the original $N \times d$ data matrix X to an $N \times 2d$ dimensional one that we denote as $\widehat{X}$, with its columns $\widehat{X}^j$ being the original X^j for $j = 1, \ldots, d$, and $\widehat{X}^j = -X^{j-d} = -\widehat{X}^{j-d}$ for $j = d+1, \ldots, 2d$. Setting $\alpha_j = \tilde{\beta}_j$ if $\tilde{\beta}_j > 0$, $\alpha_{d+j} = -\tilde{\beta}_j$ if $\tilde{\beta}_j < 0$ and the remaining $\alpha_k = 0$, we have $X\tilde{\beta} = \widehat{X}\alpha$ and

$$\left\| \tilde{y} - X\tilde{\beta} \right\|_2^2 = \left\| \tilde{y} - \widehat{X}\alpha \right\|_2^2 . \tag{4}$$

In other words, ρ-CL is equivalent to finding the point nearest to $\tilde{y}$ in the convex hull $\mathcal{C}$ spanned by $(X^1, \ldots, X^d, -X^1, \ldots, -X^d)$. Thus, solving ρ-CL is equivalent to solving the Nearest Point Problem (NPP) between the N-dimensional convex hull $\mathcal{C}$ and the singleton $\{\tilde{y}\}$.

In turn, setting $\nu = 2/(2d+1)$, this NPP problem can be scaled [11] into an equivalent linear ν-SVM problem over the sample $\mathcal{S} = \{\mathcal{S}_+, \mathcal{S}_-\}$, where $\mathcal{S}_+ = \left\{X^1, \ldots, X^d, -X^1, \ldots, -X^d\right\}$ and $\mathcal{S}_- = \{\tilde{y}\}$, which can then be solved using the LIBSVM library [12]. We get the optimal NPP α^{o} from the ν-SVM optimal solution γ^{o} by scaling it back as $\alpha^{\text{o}} = (2d+1)\gamma^{\text{o}}$.

Finally the ρ-CL solution β_ρ is obtained as $(\beta_\rho)_j = \alpha_j - \alpha_{j+d}$. In particular, observe that the NPP or ν-SVM problems do not have an unique solution, something to be expected as the kernel matrix K of the ν-SVM problem is only semidefinite positive (see (8) in Sect. 4.3). On the other hand, a unique NPP solution could be achieved if we add the constraints $\alpha_j \alpha_{j+d} = 0$ for $1 \leq j \leq d$.

We finally point out that the changes for (μ, ρ)-CEN are straightforward, since we only have to add the d extra dimensions $\sqrt{\mu} e_c$ and $-\sqrt{\mu} e_c$ to the column vectors X^c and $-X^c$.

3.2. Related Work

As mentioned in Sect. 1, after M. Jaggi's observation relating Lasso to SVMs, two other papers have addressed this link. The first one is [6], that centers itself in Elastic Net and where the authors reduce constrained Elastic Net to a squared hinged loss SVM which we discuss next in a simplified form. Using our notation in (4), in [6] the $\tilde{y}$ term is moved into the $\widehat{X}$ matrix by defining $\tilde{X} = \widehat{X} - \tilde{y}\mathbf{1}^\top$, where $\mathbf{1}$ denotes the all-ones vector. One then has $\left\|\tilde{y} - \widehat{X}\alpha\right\|_2^2 = \left\|\tilde{X}\alpha\right\|_2^2$ and the problem to be solved in [6] can now be written as

$$\min_{\alpha} \left\{ \left\|\tilde{X}\alpha\right\|_2^2 + \mu \sum_1^{2d} \alpha_j^2 \right\} \quad \text{s.t.} \quad \sum_1^{2d} \alpha_j = 1 \ , \tag{5}$$

with μ the Elastic Net penalty. The authors consider then the problem

$$\min_{\beta} \left\{ \frac{1}{2}\beta^\top Q \beta + \frac{\mu}{2} \sum_1^{2d} \beta_j^2 - \sum_1^{2d} \beta_j \right\} \quad \text{s.t.} \quad \beta_j \geq 0 \ , \tag{6}$$

where we write $Q = \tilde{X}^\top \tilde{X}$. Assuming that β^* is a solution of (6), it is easy to see that $\alpha^* = \frac{\beta^*}{\|\beta^*\|}$ is a solution of (5); thus, one can deduce a solution of (5) from an appropriate solver for (6). In order to be able to use an SVM solver for this, the authors in [6] introduce a two-class problem, using as patterns the $2d$ columns in $\tilde{X}$ and the labels $y^j = 1$ for $1 \leq j \leq d$ and $y^j = -1$ for $d+1 \leq j \leq 2d$. Then (6) is the dual of the following primal problem:

$$\min_{W} \left\{ \frac{1}{2}\|W\|^2 + \frac{1}{2\mu} \sum_1^{2d} \xi_j^2 \right\} \quad \text{s.t.} \ y^j W \cdot \tilde{X}^j \geq 1 - \xi_j \ ,$$

i.e., the bias free, ℓ_2-regularized, ℓ_2-loss SVM problem. Several solvers are available for this problem and the authors of [6] use their own implementation, Support Vectors for Elastic Net (SVEN), based in O. Chapelle's approach to solve ℓ_2-loss SVMs [13], which they adapt for training in multicore and/or GPU machines and that is able to solve both the lineal primal problem (when $d \gg N$) or the dual one (when $d \leq N$). The results in [6] show that the GPU implementation is faster than the multicore one and that both are usually faster than GLMNet (which in the standard implementation does not support either multicore or GPU execution). SVEN also outperforms two other Elastic Net specific algorithms, Shotgun and L1_LS.

Chapelle's approach cannot be used with standard ℓ_1-loss SVMs and the pure Lasso problem (i.e., with $\mu = 0$) is not considered directly in [6]. Some of its authors have addressed Lasso in [7], where they start at (5) with $\mu = 0$, i.e., with the problem

$$\min_{\alpha} \left\{ \left\| \tilde{X}\alpha \right\|_2^2 \right\} \quad \text{s.t.} \ \sum_{1}^{2d} \alpha_j = 1 \ , \tag{7}$$

which is the Minimum Norm Problem (MNP) for the points in the convex hull of the $\tilde{X}^j$ column vectors. MNP is a well-known classical problem in computational geometry, and in [7] is solved using essentially Wolfe's method. This approach is compared with other alternatives such as Shotgun, L1_LS and an implementation of the Least Angle Regression (LARS) algorithm of Efron *et al.*, which preceded GLMNet in time and that has been superseded by it (GLMNet is not included in the comparisons in [7]). In general, Wolfe's method performance for the MNP problem is sublinear and can be improved by other variants, such as the MDM algorithm [14]. MDM is closely related to SMO and, in fact, the MNP problem can be solved by LIBSVM just as done in our work: simply observe that MNP can be seen as a particular case of the Nearest Point Problem taking the singleton $\{\mathbf{0}\}$ as the second hull, just as we have done here with the singleton $\{\tilde{y}\}$ (note that (7) is just a translation by $-\tilde{y}$ of (4)). Alternatively, it is easy to see that (7) is the dual of the following one-class problem

$$\min_{W,\rho,\xi} \left\{ \frac{1}{2} \|W\|_2^2 - \rho + \sum_j \xi_j \right\} \quad \text{s.t.} \ W \cdot \tilde{X}^j - \rho + \xi_j \geq 0, \xi_j \geq 0 \ ,$$

which can also be solved by LIBSVM.

Summing things up, M. Jaggi's link between the Lasso and SVM problems has opened the way to consider new solvers to Lasso and, by extension, Elastic Net, which complement (or compete) with the well-established Lasso solvers, particularly, GLMNet, as of now the golden standard for solving Lasso on general problems. The approach in [6] has the considerable advantage of working on GPU architectures. In addition, it can solve either a primal or a dual SVM problem, while LIBSVM only solves the dual one. On the other hand, the SVM solver used deals with an ℓ_2-regularized, ℓ_2-loss SVM, which is equivalent to Constrained Elastic Net but not to Lasso, and GPU-based implementations of LIBSVM are starting to appear. Lasso is addressed in [7] but the Wolfe solver proposed there should be less efficient than SMO (no GLMNet comparisons are given in that paper). All in all, if an SVM solver is to be used for both Lasso and Elastic Net, it seems that the ν-SVM solver in LIBSVM may give a balanced option.

4. FISTA, GLMNet and SMO

In this section we will briefly describe the FISTA, GLMNet and SMO algorithms and discuss their complexities.

4.1. FISTA

FISTA (Fast ISTA; [4]) combines the basic iterations in ISTA (Iterative Soft-Thresholding Algorithm)

$$\beta_k = \mathrm{softh}_\gamma \left(w_k - \gamma \big((X^\top X + \mu I) w_k - X^\top y \big) \right) ,$$

where $\mathrm{softh}_\gamma (z) = \mathrm{sign}\,(z)(|z| - \gamma)_+$, $\gamma = 1/L$ and L is a Lipschitz constant for ∇e_2, with a Nesterov acceleration step:

$$\begin{aligned} w_{k+1} &= \beta_k + \frac{t_k - 1}{t_{k+1}} (\beta_k - \beta_{k-1}) , \\ t_{k+1} &= \frac{1}{2} \left(1 + \sqrt{1 + 4t_k^2} \right) . \end{aligned}$$

Assuming $X^\top X$ and $X^\top y$ are precomputed at a fixed initial cost $O(Nd^2)$, the cost per iteration of FISTA is $O(d^2)$, i.e., that of computing $((X^\top X + \mu I)w_k)$, which dominates the $O(d)$ costs of the soft-thresholding and w updates.

In any case FISTA provides a general framework for projected gradient descent, that can be used in many other problems besides Lasso. This flexibility and wide applicability implies a computational trade-off with the efficiency of problem-specific methods, that can usually take advantage of particular characteristics of the problem at hand. Thus, in this vein, it is at a disadvantage with respect to SMO and GLMnet; moreover, in [15] we have observed that FISTA needs more iterations than GLMNet or SMO to achieve the same precision on the objective function. If we add to this that the GLMNet and SMO iterations are greatly tuned for efficiency, it is our belief that, after the results for FISTA in [15], it is not competitive with them when applied to Lasso.

4.2. GLMNet

GLMNet performs cyclic coordinate subgradient descent on the Lasso cost function. If $\beta_i \neq 0$ for all i, the subgradient direction is

$$-X^\top r^\beta + \lambda \operatorname{sign}(\beta) \ ,$$

and the β_j update is given [3] as

$$\beta'_j = \operatorname{softh}_\lambda \left(\beta_j + \left(X^\top r^\beta \right)_j \right) \ ,$$

with softh again the soft-thresholding operator. Although GLMNet linear convergence is not proved in [3], it is to be expected after the work in [16], perhaps after some modifications [17].

To compute $X^\top r^\beta$ efficiently, GLMNet carefully manages the computations $X^j y$ and $X^j X^k$ needed in such a way that there is a cost $O(N(d+1))$ the first time they are performed at a given coordinate j but afterwards their costs are only $O(d)$, and a full coordinate cycle has then a cost of $O(d^2)$ that can be further reduced to $O(dm)$ if only m components of β are non zero. This is the goal of the screening methods we discuss below. We shall use the scikit-learn Python implementation of GLMNet; while it does not allow for feature screening, it has a very compact code free of calls to auxiliary functions and, thus, very efficient.

4.3. SMO

To solve the equivalent ρ-CL, we will use the ν-SVC option in LIBSVM (i.e., use `-s 1` when calling `svm-train`). Its iterations are similar than those of SMO and its complexity analysis is done in a pure SVM context in terms of the sample size N. However, and as discussed above, sample size will be $N = 2d+1$ in our particular case. Simplifying the discussion slightly, SMO has a $O(2d) = O(d)$ cost to compute the L and U indices that yield the patterns that most violate the ν-SVC KKT conditions, and another $O(d)$ cost to maintain gradient information. To this we must add at least in the initial iterations the $O(2 \cdot 2d \cdot N) = O(dN)$ cost of computing the required dot products that LIBSVM performs to build the kernel matrix row by row.

Going back to the case of our NPP problem, note that LIBSVM makes no assumptions on the particular structure of the kernel matrix, and since the number of patterns is $2d+1$, it may eventually compute $(2d+1)^2$ dot products. However, when reducing ρ-CL to NPP the kernel matrix K has the following structure:

$$K = \widehat{X}\widehat{X}^\top = \underbrace{\begin{pmatrix} X^\top X & -X^\top X & X^\top \tilde{y} \\ -X^\top X & X^\top X & -X^\top \tilde{y} \\ \tilde{y}^\top X & -\tilde{y}^\top X & \tilde{y}^\top \tilde{y} \end{pmatrix}}_{2d+1} . \tag{8}$$

As a result, we only need to compute $X^\top X$, $\tilde{y}^\top \tilde{y}$ and $X^\top \tilde{y}$ and fill the kernel matrix accordingly. In addition, the matrix $X^\top X$ is also symmetric and only half of the dot products are actually needed. In our experiments we have used the original LIBSVM, without any modification, and we have also implemented a slightly modified version, K-SVM, where we have exploited the specific structure of the kernel matrix for this problem.

It is important to note that the kernel matrix is not precomputed before the algorithm starts but instead the rows are computed only as needed, taking into account the structure of the kernel to avoid computing the same dot product twice. Newly computed rows are next cached so they can be reused if SMO selects again the same pair of multipliers at a later iteration. It is also worth mentioning that the changes to LIBSVM are minor and only involve the way SMO gets the kernel row associated to each multiplier from the cache; no other change is made in the LIBSVM code. Theoretically, these changes could perform up to 8 times less dot products compared to the naive LIBSVM implementation for this specific problem, although in practice we observe gains of about a factor of 4 at least for problems where $d \leq N$; when N is small and $d \gg N$, the dot product costs and, hence, the time savings here are also smaller and may not compensate for the overhead that K-SVM imposes when handling the reduced kernel matrix.

At first sight the cost of an SMO iteration is about twice of a GLMNet one, as in SMO two coordinates change per iteration while one does so in GLMNet. On the other hand, LIBSVM code aims to an efficient solution of a number of different SVM-related problems; in particular, calls to several functions are made at each iteration. Thus, while being by far the best general SVM solver and one of the best publicly available codes in Machine Learning, its iterations have a larger overhead than those of the GLMNet of scikit and, probably, also longer running times. It has been proved that the convergence of SMO is linear once the coefficients of the non-support and bounded vectors get to the optimal values, which is achieved in a finite number of iterations, with at least about $O(d)$ being required [11]. Convergence afterwards is usually very fast, as its usual stopping condition $\mathcal{D} < \epsilon$, with $\mathcal{D}$ a measure of the violation of the KKT conditions (see [11, Sect. 2.2]), can then be achieved in $O(\log 1/\epsilon)$ iterations.

5. Lasso Feature Screening and SVM Shrinking

5.1. Lasso Screening

As just mentioned, any Lasso algorithm will benefit from having a smaller set of active features and, starting with the work of El Ghaoui *et al.* [8], there has been quite a substantial amount of recent proposals to accelerate Lasso by screening, i.e., removing those features that will result on zero Lasso coefficients; see [18] for a good review. The main idea is to consider Lasso's dual and exploit the KKT conditions. More precisely, UL can be written as a constrained problem, namely

$$\min_{\beta,z} \left\{ \frac{1}{2} \|y - z\|_2^2 + \lambda \|\beta\|_1 \right\} \quad \text{s.t. } z = X\beta \ ,$$

and its Lagrangian dual then becomes [19]

$$\max_{\theta} \left\{ \frac{1}{2} \|y\|_2^2 - \frac{\lambda^2}{2} \left\| \theta - \frac{y}{\lambda} \right\|_2^2 \right\} \quad \text{s.t.} \ \ \left| X^j \cdot \theta \right| \leq 1 \ , \tag{9}$$

for $1 \leq j \leq d$. Now the KKT equations relate the primal and dual optimal solutions $\beta(\lambda)$, θ_λ as $y = X\beta(\lambda) + \lambda\theta_\lambda$, i.e., $\lambda\theta_\lambda$ is the optimal residual, and

$$\begin{aligned} X^p \cdot \theta_\lambda &= \operatorname{sign}\left(\beta(\lambda)\right)_p \quad \text{if } \beta(\lambda)_p \neq 0 \ , \\ X^p \cdot \theta_\lambda &\in [-1, 1] \quad \text{if } \beta(\lambda)_p = 0 \ . \end{aligned}$$

Notice that $\mathcal{F} = \{\theta : |X^p \cdot \theta| \leq 1, 1 \leq p \leq d\}$, the feasible set of (9), is a closed, convex polytope, and problem (9) can be interpreted as finding the projection $\theta_\lambda = \mathrm{P}_{\mathcal{F}}\left(y/\lambda\right)$ of y/λ on $\mathcal{F}$. The general idea in screening is to exploit this by identifying a convex region $\mathcal{R} = \mathcal{R}_\lambda \subset \mathcal{F}$ that contains θ_λ and then screen out all features p for which $\sup_{\mathcal{R}} |X^p \cdot \theta| < 1$. This can be done taking $\mathcal{R}$ to be an appropriate ball. More precisely, assume we know θ_{λ_0} for some λ_0; then, if $\lambda < \lambda_0$, the non-expansiveness of the projection operator ensures that

$$\|\theta_\lambda - \theta_{\lambda_0}\|_2 \leq \left\| \frac{y}{\lambda} - \frac{y}{\lambda_0} \right\|_2 = \left(\frac{1}{\lambda} - \frac{1}{\lambda_0} \right) \|y\|_2 \ .$$

Thus, the ball $B(\theta_{\lambda_0}, R_\lambda^{\lambda_0})$ with $R_\lambda^{\lambda_0} = (1/\lambda - 1/\lambda_0)\|y\|_2$ is an example of a screening region $\mathcal{R}_\lambda$ for any $\lambda < \lambda_0$. An extra advantage of working with balls is that their **support function** $\sigma_{B(c,R)}(z) = \max_{X \in B} X \cdot z$ has a simple analytic form [20], namely $\sigma_{B(c,R)}(z) = z \cdot c + R\|z\|_2$, and thus for $B = B(\theta_{\lambda_0}, R_\lambda^{\lambda_0})$,

$$\begin{aligned} \sup_{\theta \in B} |X^p \cdot \theta| &= \max\left\{\sigma_B(+X^p), \sigma_B(-X^p)\right\} \\ &= |X^p \cdot \theta_{\lambda_0}| + R_\lambda^{\lambda_0} \|X^p\|_2 \ . \end{aligned}$$

The slightly more complicated and more precise dome regions [18, 20] are also used for screening, as their support functions are also relatively simple to compute.

We briefly discuss now how to find a ball center θ_{λ_0}. Since $0 \in \mathcal{F}$, we have $y/\lambda \in \mathcal{F}$ for λ big enough. In fact, setting $\lambda_{\max} = \max_p |X^p \cdot y|$, it is well known that $\beta(\lambda_{\max}) = 0$, i.e., $\theta_\lambda = y/\lambda$ if $\lambda \geq \lambda_{\max}$; therefore $B(y/\lambda_{\max}, R_\lambda^{\lambda_{\max}})$ is a screening region for any $\lambda < \lambda_{\max}$. However it is easy to see [20] that a small enough λ results in an empty test for the basic SAFE procedure in [8], as no feature meets it. The same is true for the basic strong rule of [9] that for unit norm features X^j screens out those j for which $\left|X^j \cdot y\right| < 2\lambda - \lambda_{\max}$, that becomes empty if $\lambda < \lambda_{\max}/2$. Another drawback is that it may incorrectly screen out a feature, as it is also a sphere test over $B(y/\lambda, R)$ for an appropriate R but θ_λ may not lie in that sphere; see [18, Sect. 4.2.2].

The usual way to get more precise regions is to work in a regularization path setting where solutions θ_{λ_k} are successively computed over a sequence $\lambda_0 = \lambda_{\max} > \lambda_1 > \dots$ and (recursive) SAFE [8] or (sequential) strong [9] rules are applied when computing $\theta_{\lambda_{k+1}}$ once θ_{λ_k} is known. In fact, there has been a substantial number of recent contributions [21, 22, 23] with great promise and in some cases (for instance [23]) with publicly available Python–Matlab implementations. However, note that full regularization paths have to be found when looking for an optimal λ, but not so for the repeated construction of models with a previously fixed λ.

Finally, perfect screening (i.e., training Lasso with only the non-zero coefficients) leads to a cost per GLMNet iteration of $O(dm) = O(rd^2)$ with $r = m/d$ the final sparsity of the model. Obviously, screening will be better for the more sparse models induced by large λ penalties but possibly not so when the optimal λ is rather smaller.

5.2. SVM Shrinking

A similar effect can be achieved in SVM training if we screen out of the active set those patterns that won't become support vectors. Some recent papers [19, 24] also deal with non-support vector screening for SVM classifiers; in any case, we will concentrate here on the well-known shrinking technique for SMO [12]. Notice that when going from Lasso to NPP or ν-SVMs sample size of the positive class becomes $2d$ and for a non-SV X^j we have $\alpha_j = 0$; in other words, to screen out a β_j is equivalent to remove both X^j and $-X^j$ from the ν-SVM active set, i.e., to shrink these non-SVs. SMO shrinking relies on the fact that after some iteration T, the non-SVs coefficients do not change from their bound values. Thus, after that T (provided it is guessed correctly) we can reduce the active set to those X^j such that their α_j are not bounded. The LIBSVM implementation of shrinking [12, Sect. 5.1] selects non-SV candidates X^j according to whether α_j^k is bounded and the j-th gradient component of the SVM dual function at α_j^k goes "against" the bounding box constraint. This is done every $\min\{N, 1\,000\}$ iterations, i.e., $\min\{2d+1, 1\,000\}$ in our case. Of course, this is not a "safe" procedure in the Lasso terminology; to ensure it doesn't lead to a wrong solution, LIBSVM reconstructs the entire gradient once the stopping condition $\mathcal{D} \leq 10\epsilon$ holds for the shrunk active set, and a more precise shrinking criterion is then applied. This is done again when the condition $\mathcal{D} \leq \epsilon$ is met on the new shrunk set. Obviously, over very large samples (or in our case, input dimensions) for which the kernel matrix does not fit into memory, gradient reconstruction is very expensive and SMO training with shrinking may actually be costlier. In LIBSVM [12, Sect. 5.6] the sizes of the set $\mathcal{F} = \{j : 0 < \alpha_j < \nu\}$ of unbounded coefficients and of the active set $\mathcal{A}$ are compared and a warning is issued if $2|\mathcal{F}| < |\mathcal{A}|$.

Shrinking will reduce the iteration cost of SMO, $O(2d)$, by a fraction $r \simeq m/2d$, with $m = |\mathcal{A}|$, the size of $\mathcal{A}$, i.e., the number of non-zero features in the final α solution. However, since this solution is not unique, the α sparsity does not have to coincide with that of the Lasso solution: just notice that a fully sparse $\beta = 0$ Lasso solution can be derived from a fully non-sparse $\alpha_j = 1/2d$, $\alpha_{j+d} = -1/2d$ NPP solution. Thus, here screening would be computationally very effective but shrinking would not help at all. On the other hand, a fully non-sparse Lasso may correspond with a NPP solution with a sparsity fraction $r = 1/2$; thus, here screening would not help but shrinking would reduce the cost of SMO iterations by half.

In any case, in all our experiments Lasso and ν-SVM sparsity coincide; this is probably due to SMO's initialization in LIBSVM, which is done [12, Sect. 4.2.3] giving to the multipliers of the first $\nu N/2$ elements of the positive and negative samples the initial value of 1. Since here sample size N is just $2d+1$

and $\nu = 2/(2d+1)$, SMO will simply set $\alpha_1 = 1$ and all other multipliers equal to 0. In other words, the initial solution is as sparse as possible and this seems to be carried on to the final solution.

6. Numerical Experiments

6.1. Datasets and Methods

We compare next the performance of GLMNet and LIBSVM when solving equivalent λ-UL and ρ-CL problems in eleven datasets. Of these, eight correspond to regression problems: `prostate`, used in the original Lasso paper [1], `housing`, `year`, `ctscan` and `cpusmall` from the UCI repository, `trajectory`, from the Machine Learning Dataset Repository, `ree`, the problem of predicting wind energy production over peninsular Spain using numerical weather forecasts, and `mnist_reg`, a regression problem built from the MNIST dataset. In this last problem we proceed as in [18], randomly selecting 500 feature images from each digit as well as a random digit target image and building then 28×28 samples with feature dimension 5 000 by pairing the i, j pixels of the 5 000 feature images with the i, j pixel of the target image. The `ree` features correspond to 8 weather variable forecasts at each point of a 57×35 rectangular grid that encompasses the Iberian peninsula. In addition to the previous regression datasets, we also considered three classification datasets for biomedical problems: `leukemia`, `colon_cancer` and `breast_cancer`, from the LIBSVM dataset repository. These two-class datasets are transformed to a regression problem by simply predicting the value of the -1/1 class label. The reason for this addition is to further explore the case where the number of features is much greater than the number of patterns, which is easier to find in classification problems from the medical domain.

Table 1 summarizes in columns 4 and 5 the sample sizes and input dimensions of all the previous datasets. It is worth mentioning that all of them correspond to real-world problems with a wide variety of sizes. Some of them are also quite

Table 1: Initial and scaled λ values, sample sizes and input dimensions of the datasets considered.

Dataset	Optimal λ	$\lambda/\lambda_{\max}$	Size	Dim.
`prostate`	2.035×10^{-3}	0.002 8	67	8
`housing`	8.186×10^{-3}	0.011 2	378	13
`year`	4.534×10^{-4}	0.002 1	46 215	90
`ctscan`	3.748×10^{-3}	0.006 8	53 500	385
`cpusmall`	1.631×10^{-2}	0.047 9	6 143	12
`mnist_reg`	7.760×10^{-3}	0.036 4	784	5 000
`trajectory`	1.021×10^{-2}	0.129 2	20 000	298
`ree`	1.290×10^{-1}	0.164 1	5 698	15 960
`leukemia`	5.294×10^{-3}	0.497 0	72	7 129
`colon_cancer`	6.191×10^{-3}	0.071 1	62	2 000
`breast_cancer`	3.537×10^{-2}	0.347 0	44	7 129

big for a regression setting, either in number of patterns (`ctscan`, `year`) or input dimension (`mnist_reg`, `ree`, `leukemia`, `breast_cancer`).

As it is well known, training complexity greatly depends on λ. We will consider three possible λ values for UL: an optimal λ^* obtained according to the regularization path procedure of [3], a smaller $\lambda^*/2$ value which should result in longer training and possibly a smaller sparsity, and a stricter penalty $2\lambda^*$ value with the opposite effect. As discussed in [18], the ratio $\lambda/\lambda_{\max}$ is invariant to scaling and is thus a better measure of the regularization strength being applied in Lasso. The optimal λ^* and their $\lambda_{\max}$ scalings for the problems considered are given in the second and third columns of Table 1. Most problems have small scaled λ values; they are higher and close to 0.5 in `leukemia` and `breast_cancer`. The corresponding ρ parameters are computed as $\rho = \|\beta(\lambda)\|_1$, with $\beta(\lambda)$ the optimal solution for λ-UL.

To make a balanced comparison, for each λ and dataset we first make a long run of a given algorithm M so that it converges to a $\beta^*_M(\lambda)$ that we take as its optimum. We then compare for each M the evolution of $f_M(\beta^k_M(\lambda)) - f_M(\beta^*_M(\lambda))$, with $\beta^k_M(\lambda)$ the coefficients at the k-th iteration of algorithm M, until it is smaller than a threshold ϵ that we fix as 10^{-6} for all datasets.

As mentioned, we will use the scikit-learn implementations of GLMNet and of ν-SVM, in itself a wrapper over the LIBSVM implementation, to which we add the modified LIBSVM code described in Sect. 4.3 to reduce the number of kernel operations. We denote these algorithms as GLMNet, SVM and K-SVM respectively. Note that all methods have a compiled C core, so we may expect time comparisons to be broadly homogeneous. As mentioned before, we do not give results for FISTA.

6.2. Iterations and Running Times

All the experiments were run in an Intel(R) Xeon(R) server with 16 E5-2680 2.70GHz CPUs and 128 Gb of RAM. Table 2 shows in columns 3 to 5 the number of iterations that GLMNet, SVM and K-SVM respectively require to arrive at the $\epsilon = 10^{-6}$ threshold and in columns 6 to 8 the times needed to achieve the same precision. We consider GLMNet and SVM iterations as equivalent: even though GLMNet only changes one coefficient per iteration and SVM two, it also works on a $2d$-dimensional space when GLMNet does it in a d-dimensional one. The table shows that K-SVM beats GLMNet for all datasets and λ values except for the `trajectory` problem. Besides, for the regression datasets, K-SVM improves SVM running times by a factor that approximately lies between 2 and 4, while both perform the same number of iterations in all problems, as it should be. However, for the classification datasets, SVM is faster than K-SVM. As mentioned before, we believe this is due to these datasets having a very small number of patterns and dimensions between 40 and 100 times larger; therefore the dot products computed are very cheap while the kernel matrix is rather large. This, combined with the fact that the optimization finishes in very few iterations (note that at most two dot products are computed in each iteration), makes the overhead of taking into account the kernel matrix structure to be worse than just computing the full dot products. As a conclusion, SVM may be better suited for problems where the ratio p/N is very big. On the other hand, for the three such classification datasets both SVM and K–SVM are faster than

Table 2: Sparsity of final solutions, number of iterations and running times.

		Iterations			Time (ms)		
Dataset	Spars.	GLMNet	SVM	K-SVM	GLMNet	SVM	K-SVM
`prostate`	8/8	102	40	40	0.051	0.030	0.023
	8/8	103	35	35	0.052	0.026	0.020
	7/8	93	28	28	0.048	0.023	0.017
`housing`	12/13	607	71	71	0.567	0.160	0.073
	12/13	594	53	53	0.555	0.152	0.064
	11/13	216	33	33	0.204	0.143	0.053
`year`	90/90	5 153	693	693	456.753	757.949	182.591
	89/90	5 148	559	559	451.329	731.710	176.815
	85/90	4 998	445	445	425.374	718.278	175.916
`ctscan`	273/385	101 130	944	944	8 371.474	11 334.768	3 707.740
	226/385	78 710	629	629	5 949.542	9 316.175	3 354.670
	165/385	53 630	387	387	3 488.810	6 755.625	2 702.005
`cpusmall`	9/12	135	21	21	1.126	1.363	0.366
	8/12	144	13	13	1.150	1.201	0.338
	6/12	45	11	11	0.337	0.896	0.308
`mnist_reg`	40/5 000	1 040 395	87	87	1 451.346	371.682	217.461
	22/5 000	635 452	42	42	940.499	185.491	106.909
	13/5 000	550 356	24	24	694.931	109.587	63.693
`trajectory`	45/297	8 555	164	164	228.461	583.402	264.339
	17/297	2 588	32	32	61.546	253.426	108.617
	6/297	1 313	9	9	28.673	84.417	36.852
`ree`	73/15 960	1 784 764	185	185	9 949.901	17 033.588	9 386.198
	43/15 960	1 385 271	108	108	7 058.281	8 691.214	5 078.536
	23/15 960	1 114 658	52	52	5 541.830	5 097.402	2 789.337
`leukemia`	28/7 129	415 227	71	71	197.762	49.557	87.672
	20/7 129	225 846	28	28	58.096	26.253	50.891
	8/7 129	115 898	9	9	60.959	10.155	22.217
`colon_cancer`	47/2 000	184 353	469	469	88.677	31.467	63.017
	33/2 000	101 231	227	227	45.856	19.183	38.180
	25/2 000	62 377	54	54	29.894	7.411	14.476
`breast_cancer`	29/7 129	219 503	100	100	97.303	50.856	92.861
	16/7 129	107 397	27	27	47.075	23.024	40.219
	7/7 129	59 597	8	8	26.265	5.919	16.873

GLMNet in our experiments.

Problems with a large dimension, such as `mnist_reg`, `ree` and the classification datasets, do need very few SVM iterations. This is related to the sparsity of the final solutions, that is given in the second column of Table 2, in the form s/d, with s the number of non-zero coefficients and d the problem dimension. As mentioned, the Lasso and SVM solutions might have different numbers of non-zero coefficients; however, in our experiments, both are the same and so we present the sparsity values in just one column. In particular, it seems that SVM does not introduce artificial non-zero coefficients. This is probably due in part to the initialization of SMO in LIBSVM, that sets all coefficients but one to zero. As the table shows, the sparsities in `mnist_reg`, 22 non-zero coefficients out of 5 000 for λ^*, or `ree`, 43 out of 15 960, are very large. In the `mnist_reg` case this is due to the fact that many features correspond to zero pixels and also that only 1/10 of the features correspond to images from the digit that we are trying to predict. In `ree` it is possibly due to the large correlation between weather variables at nearby grid points. The classification datasets also have a very big sparsity factor, probably because the ratio p/N is huge and therefore there are many irrelevant variables, as is usually the case in medical data.

This behavior is further illustrated in Fig. 1 that depicts for the λ^* penalty the evolution of running times until the ϵ threshold is reached. In the small sample regression problems of `housing`, `prostate` and `mnist_reg`, where kernel operations are less costly, the running time of SVM is smaller than that of GLMNet even for rather modest values of the objective function (about 10^{-3}), but consistently larger in all the others. On the other hand, K-SVM reaches a value of 10^{-3} on the objective function faster than GLMNet in `prostate`, `housing`, `mnist_reg` and `cpusmall`, a value of 10^{-4} in `year`, and a value of 10^{-5} in `ctscan` and `ree`. GLMNet is the clear winner in `trajectory`.

Fig. 1 shows the same evolution for the three classification problems. As mentioned, in this setting where $p \gg N$ there are very few kernel operations (because of the small number of iterations until convergence) and they are also very cheap (because of a very small N), so the SVM without modifications

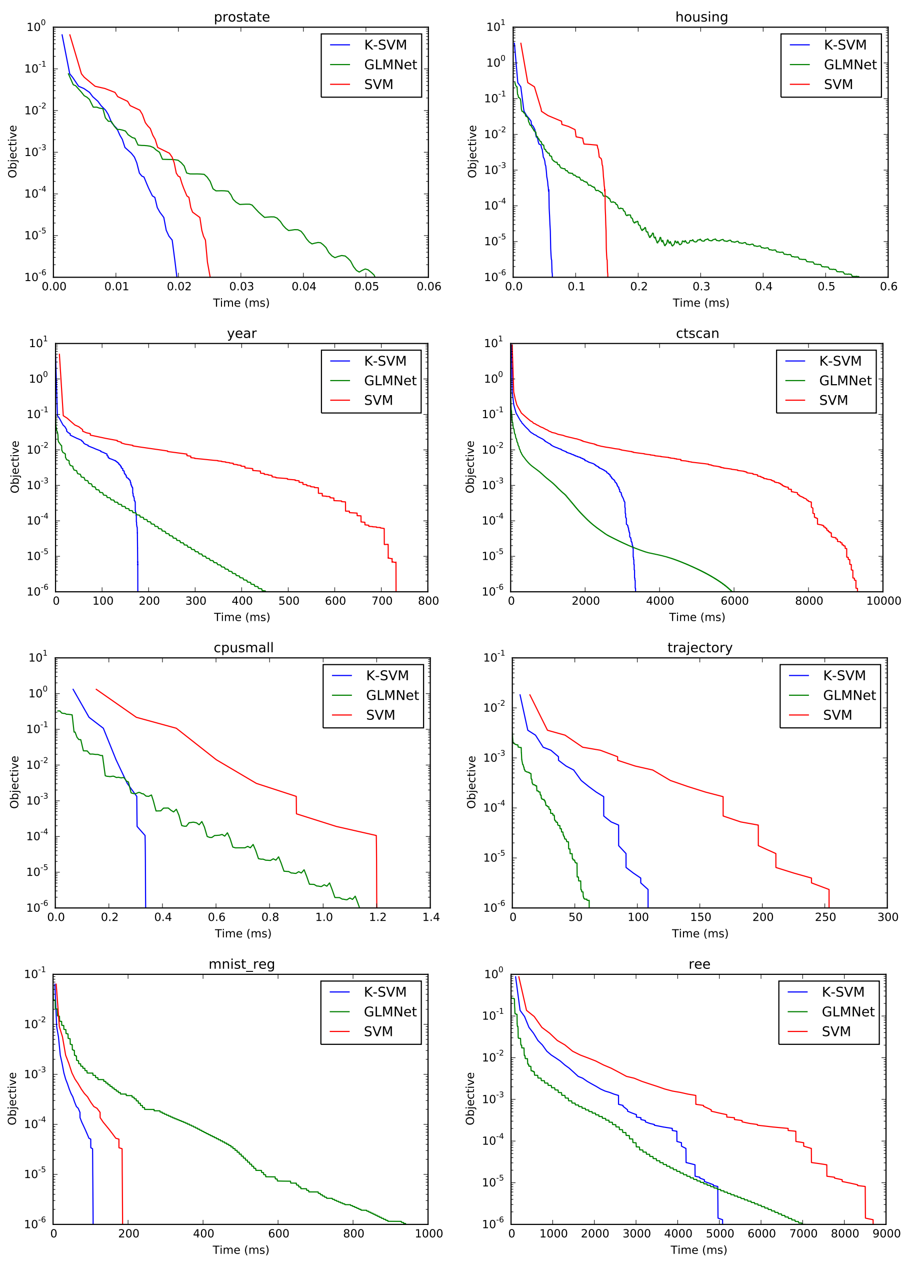

Figure 1: Time evolution of the objective function for the eight regression datasets with λ^* as the penalty factor.

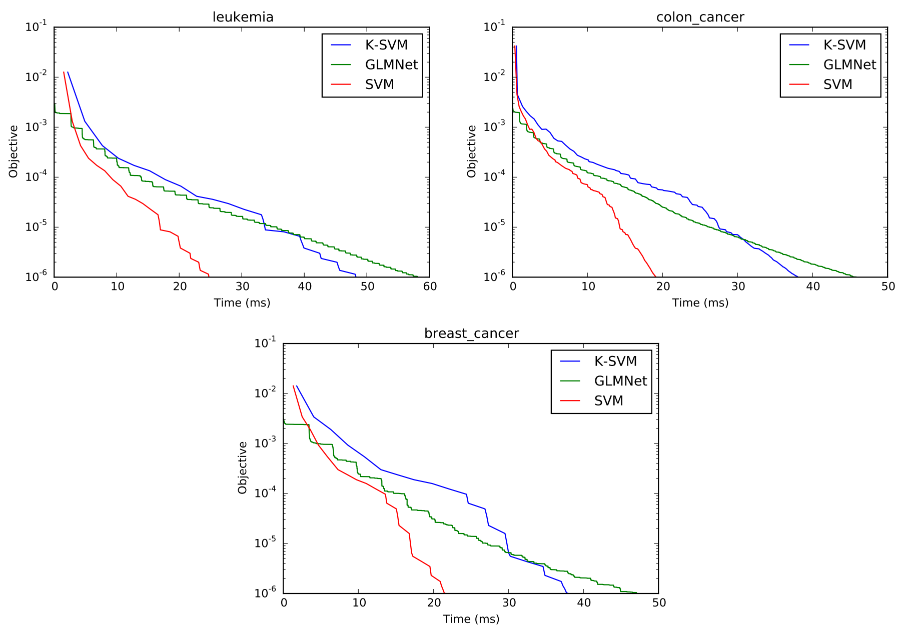

Figure 2: Time evolution of the objective function for the three classification datasets with λ^* as the penalty factor.

performs very well. Even though they are very sparse problems, the inner loop of GLMNet still has to go through all coefficients and thus the number of iterations for this algorithm is still very high. These problems could probably benefit from some kind of feature screening which is not yet implemented in scikit-learn GLMNet, although as discussed in Sect. 5, the ratio $\lambda/\lambda_{\max}$ is < 0.5 which implies no gain when using the strong rules and probably not a large one for other screening procedures. In addition, it is important to emphasize that **the shrinking feature of LIBSVM was also disabled** for all the regression and classification experiments, so we believe the comparisons reported to be fair.

Even if we have not used it, recall that LIBSVM only applies shrinking after the first $\min\{2d + 1, 1\,000\}$ iterations. Comparing the dimensions in Table 1 and the number of iterations in Table 2, this implies that it will have no effect on the large dimensional `ctscan`, `trajectory`, `mnist_reg`, `ree` and classification

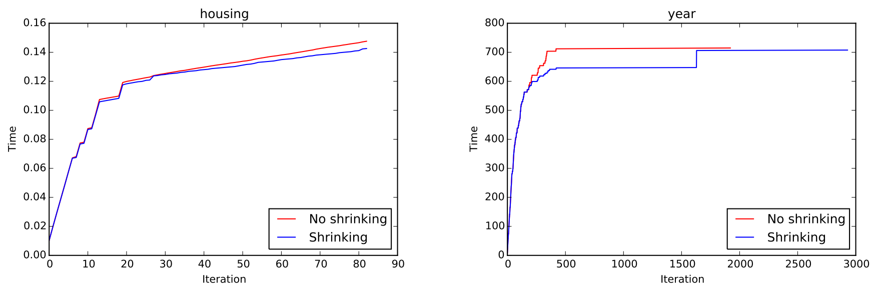

Figure 3: Time versus iterations for the `housing` and `year` datasets with penalty $2\lambda^*$ with and without shrinking.

datasets. In the remaining datasets, we believe shrinking would have little to no effect in `prostate` and `cpusmall`, most likely because the gradient conditions for shrinking are not met, and a very small one in `housing` and `year`. The evolution of training times versus iterations in these problems with and without shrinking is given for the stronger sparsity inducing $2\lambda^*$ case in Fig. 3; in them we have used the standard version of LIBSVM. As it can be seen, the effect is very modest, something to be partially expected given the small sparsity of the optimal solutions.

7. Discussion and Conclusions

M. Jaggi's recently observed that ρ-CL can be broadly reduced to an SVM problem, opening the way to the application of SVM training methods as alternatives to GLMNet, a procedure that takes great advantage of the particular structure of Lasso and that can be considered the current state of the art to solve the Lasso problem. Here we have made that reduction more precise showing that ρ-CL can be reformulated as a Nearest Point Problem which, in turn, can be further reduced to ν-SVC, a variant of standard SV classification. The straight use of the ν-SVC option of the well-known and widely used LIBSVM library is competitive with GLMNet in six of our eleven problems. When it is not, its running times remain close and, moreover, the particular form of the kernel matrix of the equivalent ν-SVC problem makes clear how to adapt the handling of kernel operation of LIBSVM to achieve further time gains. Following this path we have experimentally shown that this small modification of LIBSVM results in a procedure that is faster than GLMNet in all of the problems considered except `trajectory`.

Recently several screening procedures have been proposed to speed up solving Lasso by reducing the size of the active feature set. While very elegant and powerful on concrete setups, screening procedures face two drawbacks. First, one usually needs to know the solution of Lasso $\beta(\lambda_0)$ on a previous λ_0 penalty when applying screening over a new $\lambda < \lambda_0$; this can be done advantageously on a sequential, regularization path setting while looking for an optimal λ but less so when that λ is already fixed and Lasso solutions have to be repeatedly computed over it. The second drawback is that screening will reduce running times more effectively the sparser a problem is. In fact, for the datasets considered in [18], screening in sequential strong Lasso only reduces training times for penalty ratios $\lambda/\lambda_{\max}$ larger than 0.5. Some of the sequential methods in [18] offer in some problems time reductions for ratios as small as 0.05; in all our experiments this ratio (computed for a λ^* optimal for the problem at hand) was below 0.165, in six of them below 0.05 and in four below 0.02. Besides, high sparsity usually corresponds to large λ penalties but, on the other hand, large penalties may result in poorer models; again, in our problems the optimal λ values derived using the regularization path procedure of [3] yielded quite small penalty ratios.

Although different in many aspects, shrinking is a natural counterpart to screening when training SVMs. We have not used it in our comparisons with GLMNet but have independently observed that it would have had only slight effects on just two datasets. This may be partially due to the conservative way shrinking is applied in LIBSVM. A conservative approach is certain sensible when SVMs are to be trained using non-linear kernels such as Gaussian ones, since shrinking is not a safe procedure and it may initially lead to wrong solutions and a very costly reconstruction of the entire SVM gradient afterwards. On the other hand, a more aggressive shrinking could be safer in the linear

SVM problems that arise from the ρ-CL reduction and thus effectively reduce training times. Another way to speed up Lasso reductions to SVM could be to replace LIBSVM with the LIBLINEAR library [25], specifically tailored to linear homogeneous SVMs; however, as of now ν-SVMs is not included among the problems LIBLINEAR solves.

In any case, and besides the references above, we also point out that there has been lately a large research effort dealing with coordinate descent methods for several problems, with Lasso among them (recall that GLMNet is also a coordinate descent method). Among the many relevant papers (see [26] for a review up to 2012) we can mention [27, 28] as good examples of the techniques considered when dealing (as we do) with single CPU algorithms. But, as mentioned, work is being increasingly done extending and parallelizing SVM solvers so that they can take advantage of the many recent hardware advances in multiple core CPU and general purpose graphics processing units (GPUs). An empirical analysis of SVM parallelization for multi-core CPUs and GPU architectures is in [29] and such settings are exploited in [6]. Parallelization requirements often entail working with specific solvers that may introduce problem simplifications (such as the homogeneous model assumption of LIBLINEAR) which, in turn, may require extra coding modifications so that ν-SVM fits in them. However, the analysis in [29] suggests that simpler, implicit approaches to SVM parallelism may result in substantial computational gains with a less costly programming effort, which could be immediately exploited for a more efficient solution for Lasso. From the more general point of view of distributed coordinate descent, the papers [30, 31] are particularly noteworthy. In contrast with single CPU algorithms, these papers propose distributed multicore coordinate descent methods which rely on very clever sub-block sampling techniques and can handle very efficiently data matrices with 2×10^9 rows, 10^9 features and up to 20×10^9 non-zero entries. Clearly, they will beat any single CPU algorithm.

Finally, we observe that the Lasso and related problems receive a constant attention in many application areas [32, 33, 34] and, moreover, they are at the core of many other problems in convex regularized learning, such as Fused

Lasso, wavelet smoothing or trend filtering, currently solved using specialized algorithms. A ν-SVC approach could provide for them the same faster convergence that we have illustrated here for standard Lasso. We are working on these and other related questions.

Acknowledgments

With partial support from Spain's grants TIN2013-42351-P, TIN2016-76406-P, TIN2015-70308-REDT and S2013/ICE-2845 CASI-CAM-CM; work also supported by project FACIL–Ayudas Fundación BBVA a Equipos de Investigación Científica 2016 and the UAM–ADIC Chair for Data Science and Machine Learning. The first author is also supported by the FPU–MEC grant AP-2012-5163. We gratefully acknowledge the use of the facilities of Centro de Computación Científica (CCC) at UAM and thank Red Eléctrica de España for kindly supplying wind energy data.

References

[1] R. Tibshirani, Regression Shrinkage and Selection Via the Lasso, Journal of the Royal Statistical Society and Series B 58 (1994) 267–288.
URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.7574`

[2] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67 (2) (2005) 301–320.
URL `http://dx.doi.org/10.1111/j.1467-9868.2005.00503.x`

[3] J. H. Friedman, T. Hastie, R. Tibshirani, Regularization Paths for Generalized Linear Models via Coordinate Descent, Journal of Statistical Software 33 (1) (2010) 1–22.
URL `http://www.jstatsoft.org/v33/i01`

[4] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Img. Sci. 2 (1) (2009) 183–202.
URL `http://dx.doi.org/10.1137/080716542`

[5] M. Jaggi, An equivalence between the lasso and support vector machines, in: J. A. K. Suykens, M. Signoretto, A. Argyriou (Eds.), Regularization, optimization, kernels, and support vector machines, Chapman and Hall/CRC, 2014, pp. 1–26.

[6] Q. Zhou, W. Chen, S. Song, J. R. Gardner, K. Q. Weinberger, Y. Chen, A reduction of the elastic net to support vector machines with an application to GPU computing, in: AAAI, 2015, pp. 3210–3216.

[7] Q. Zhou, S. Song, G. Huang, C. Wu, Efficient lasso training from a geometrical perspective, Neurocomputing 168 (2015) 234–239. `doi:10.1016/j.neucom.2015.05.103`.
URL `http://dx.doi.org/10.1016/j.neucom.2015.05.103`

[8] L. E. Ghaoui, V. Viallon, T. Rabbani, Safe feature elimination in sparse supervised learning, CoRR abs/1009.4219.
URL `http://arxiv.org/abs/1009.4219`

[9] R. Tibshirani, J. Bien, J. Friedman, T. Hastie, N. Simon, J. Taylor, R. J. Tibshirani, Strong rules for discarding predictors in lasso-type problems, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 74 (2) (2012) 245–266.

[10] S. Boyd, L. Vandenberghe, Convex optimization, Cambridge university press, 2004. `doi:10.1017/cbo9780511804441`.

[11] J. López-Lázaro, J. R. Dorronsoro, Linear convergence rate for the MDM algorithm for the nearest point problem, Pattern Recognition 48 (4) (2015) 1510–1522.
URL `http://dx.doi.org/10.1016/j.patcog.2014.10.015`

[12] C.-C. Chang, C.-J. Lin, Libsvm: A library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 27:1–27:27. doi:10.1145/1961189.1961199.
URL http://doi.acm.org/10.1145/1961189.1961199

[13] O. Chapelle, Training a support vector machine in the primal, Neural Computation 19 (5) (2007) 1155–1178. doi:10.1162/neco.2007.19.5.1155.
URL http://dx.doi.org/10.1162/neco.2007.19.5.1155

[14] A. Torres, J. R. Dorronsoro, Conjugate descent for the minimum norm problem, in: OPT2015: Optimization for Machine Learning, NIPS 2015 Workshop, 2015.
URL http://opt-ml.org/papers/OPT2015_paper_29.pdf

[15] C. M. Alaíz, A. Torres, J. R. Dorronsoro, Solving constrained lasso and elastic net using ν-svms, in: Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN 2015, Presses universitaires de Louvain, 2015, pp. 267–272.

[16] S. Yun, P. Tseng, K. Toh, A block coordinate gradient descent method for regularized convex separable optimization and covariance selection, Math. Program. 129 (2) (2011) 331–355.
URL http://dx.doi.org/10.1007/s10107-011-0471-1

[17] G.-X. Yuan, C.-H. Ho, C.-J. Lin, An improved GLMNET for l1-regularized logistic regression, Journal of Machine Learning Research 13 (2012) 1999–2030.
URL http://www.csie.ntu.edu.tw/~cjlin/papers/l1_glmnet/long-glmnet.pdf

[18] Z. J. Xiang, Y. Wang, P. J. Ramadge, Screening tests for lasso problems, IEEE Transactions on Pattern Analysis & Machine Intelligence (to appear).
URL http://doi.ieeecomputersociety.org/10.1109/TPAMI.2016.2568185

[19] J. Wang, B. Lin, P. Gong, P. Wonka, J. Ye, Lasso screening rules via dual polytope projection, CoRR (2012) –1–1.

[20] O. Fercoq, A. Gramfort, J. Salmon, Mind the duality gap: safer rules for the lasso, in: ICML, 2015.
URL `http://arxiv.org/pdf/1505.03410v1`

[21] J. Wang, B. Lin, P. Gong, P. Wonka, J. Ye, Lasso screening rules via dual polytope projection, CoRR (2012) –1–1.

[22] J. Liu, Z. Zhao, J. Wang, J. Ye, Safe screening with variational inequalities and its applicaiton to LASSO, CoRR abs/1307.7577.
URL `http://arxiv.org/abs/1307.7577`

[23] A. Bonnefoy, V. Emiya, L. Ralaivola, R. Gribonval, Dynamic screening: Accelerating first-order algorithms for the lasso and group-lasso, IEEE Transactions on Signal Processing 63 (19) (2015) 5121–5132. `doi:10.1109/TSP.2015.2447503`.

[24] K. Ogawa, Y. Suzuki, I. Takeuchi, Safe screening of non-support vectors in pathwise SVM computation, in: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013, 2013, pp. 1382–1390.

[25] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: a library for large linear classification, Journal of Machine Learning Research 9 (2008) 1871–1874.
URL `http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf`

[26] G. Yuan, C. Ho, C. Lin, Recent advances of large-scale linear classification, Proceedings of the IEEE 100 (9) (2012) 2584–2603. `doi:10.1109/JPROC.2012.2188013`.
URL `http://dx.doi.org/10.1109/JPROC.2012.2188013`

[27] R. Tomioka, T. Suzuki, M. Sugiyama, Super-linear convergence of dual augmented lagrangian algorithm for sparsity regularized estimation, Jour-

nal of Machine Learning Research 12 (2011) 1537–1586.
URL http://dl.acm.org/citation.cfm?id=2021050

[28] S. Shalev-Shwartz, T. Zhang, Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization, in: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014, 2014, pp. 64–72.
URL http://jmlr.org/proceedings/papers/v32/shalev-shwartz14.html

[29] S. Tyree, J. R. Gardner, K. Q. Weinberger, K. Agrawal, J. Tran, Parallel support vector machines in practice, CoRR abs/1404.1066.
URL http://arxiv.org/abs/1404.1066

[30] P. Richtárik, M. Takáč, Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function, Math. Program. 144 (1-2) (2014) 1–38. doi:10.1007/s10107-012-0614-z.
URL http://dx.doi.org/10.1007/s10107-012-0614-z

[31] P. Richtárik, M. Takáč, Parallel coordinate descent methods for big data optimization, Math. Program. 156 (1-2) (2016) 433–484. doi:10.1007/s10107-015-0901-6.
URL http://dx.doi.org/10.1007/s10107-015-0901-6

[32] D. Vidaurre, C. Bielza, P. Larrañaga, Classification of neural signals from sparse autoregressive features, Neurocomputing 111 (2013) 21 – 26. doi:http://dx.doi.org/10.1016/j.neucom.2012.12.013.
URL http://www.sciencedirect.com/science/article/pii/S0925231213000271

[33] B. Xu, K. Huang, I. King, C.-L. Liu, J. Sun, N. Satoshi, Graphical lasso quadratic discriminant function and its application to character recognition, Neurocomputing 129 (2014) 33 – 40. doi:http://dx.doi.org/10.1016/j.neucom.2012.08.073.

URL http://www.sciencedirect.com/science/article/pii/S0925231213009843

[34] Z. Ren, Y. Yang, F. Bao, Y. Deng, Q. Dai, Directed adaptive graphical lasso for causality inference, Neurocomputing 173, Part 3 (2016) 1989 – 1994. doi:http://dx.doi.org/10.1016/j.neucom.2015.08.032.
URL http://www.sciencedirect.com/science/article/pii/S0925231215011856