From safe screening rules to working sets for faster Lasso-type solvers

Mathurin Massias mathurin.massias@inria.fr
INRIA, Université Paris Saclay, Palaiseau, France
Alexandre Gramfort alexandre.gramfort@inria.fr
INRIA, Université Paris Saclay, Palaiseau, France
Joseph Salmon joseph.salmon@telecom-paristech.fr
LTCI, Telecom ParisTech, Université Paris-Saclay, Paris, France

Abstract

Convex sparsity-promoting regularizations are ubiquitous in modern statistical learning. By construction, they yield solutions with few non-zero coefficients, which correspond to saturated constraints in the dual optimization formulation. Working set (WS) strategies are generic optimization techniques that consist in solving simpler problems that only consider a subset of constraints, whose indices form the WS. Working set methods therefore involve two nested iterations: the outer loop corresponds to the definition of the WS and the inner loop calls a solver for the subproblems. For the Lasso estimator a WS is a set of features, while for a Group Lasso it refers to a set of groups. Here we show that the Gauss-Southwell rule (a greedy strategy for block coordinate descent techniques) leads to fast solvers in this case. Combined with a working set strategy based on an aggressive use of so-called Gap Safe screening rules, we propose a solver achieving state-of-the-art performance on sparse learning problems. Results are presented on Lasso and multi-task Lasso estimators.

1 Introduction

Sparsity-promoting regularization has had a considerable impact on high dimensional statistics both in terms of applications and on the theoretical side [4]. Yet they come with a cose, since their use requires solving high-dimensional constrained or non-smooth optimization problems, for which dedicated advanced solvers are necessary [1].

Various optimization strategies have been proposed to accelerate the solvers for problems such as Lasso or sparse logistic regression involving ℓ_1 regularization, multi-task Lasso, multinomial logistic or group-Lasso involving ℓ_1/ℓ_2 mixed-norms [18, 11, 7]. We will refer to these problems as Lasso-type problems [1]. For these, so-called (block) coordinate descent (BCD) techniques [28, 7, 31, 23], which consist in updating one coordinate or one block of coordinates at a time, have had massive success. Different BCD strategies exist depending on how one iterates over coordinates: cyclic rule [7], random [23], or greedy [24, 31]. The later rule, recently studied by [29, 17, 21] is historically known as the Gauss-Southwell (GS) rule [26].

To scale up generic solvers, one recurrent idea has been to limit the size of the problems solved. This idea is at the heart of the so-called *strong rules* [27], but similar ideas can be found earlier in the Lasso literature [22, 12, 13] and also more recently for example in the BLITZ method [9, 10]. In parallel of these WS approaches where a BCD solver is run many times, first on a small subproblem then on growing ones, it has been proposed to employ so called *safe rules* [5]. While a WS algorithm starts a BCD solver using a subset of features, eventually ignoring good ones that shall be later considered, safe rules discard (once and for all) from the full problem some features that are guaranteed to be inactive at convergence. The most recent versions, called Gap Safe rules, have been applied to a wide range of Lasso-type problems [6, 14, 15].

The main contributions of this paper are 1) the introduction of a WS strategy based on an aggressive use of Gap Safe rules, and 2) the demonstration that Gauss-Southwell rules combined with precomputation of Gram matrices can be competitive for the (small) subproblems when looking at running time, and not just in terms of (block) coordinate updates/epochs as previously done in the literature [17, 25].

The paper is organized as follows: in Section 2, we present how Gap Safe rules can lead to a WS strategy. We then explain how the Gauss-Southwell rule can be employed to reduce computations. Section 4 presents

OPTML 2017: 10th NIPS Workshop on *Optimization for Machine Learning* (NIPS 2017).

numerical experiments on simulations for GS based inner-solvers, and report time improvements compared to the present state-of-the-art on real datasets.

Model and notation

We denote by $[d]$ the set $\{1, \dots, d\}$ for any integer $d \in \mathbb{N}$. For any vector $u \in \mathbb{R}^d$ and $\mathcal{C} \subset [d]$, $(u)_\mathcal{C}$ is the vector composed of elements of u whose index lies in $\mathcal{C}$, and $\bar{\mathcal{C}}$ is the complementary set of $\mathcal{C}$ in $[d]$. We denote by $\mathcal{S}^r_\mathrm{B} \subset [p]$ the row support of a matrix $\mathrm{B} \in \mathbb{R}^{p \times q}$. Let n and $p \in \mathbb{N}$ be respectively the number of observations and features and $X \in \mathbb{R}^{n \times p}$ the design matrix. Let $Y \in \mathbb{R}^{n \times q}$ be the observation matrix, where q stands for the number of tasks or classes considered. The Euclidean (resp. Frobenius) norm on vectors (resp. matrices) is denoted by $\|\cdot\|$ (resp. $\|\cdot\|_F$, and the j-th row (resp. k-th column) of B by $\mathrm{B}_{j,:}$ (resp. $\mathrm{B}_{:,k}$). The row-wise $\ell_{2,1}$ group-norm of a matrix B is written $\|\mathrm{B}\|_{2,1} = \sum_j \|\mathrm{B}_{j,:}\|$. The dual norm of $\|\cdot\|_{2,1}$ norm is the ℓ_∞/ℓ_2 norm $\|\mathrm{B}\|_{2,\infty} = \max_j \|\mathrm{B}_{j,:}\|$. We denote by $\|\mathrm{B}\|_{2,0}$ the number of non-zero rows of B.

The estimator that we consider from now on is defined as a solution of the (primal) problem

$$\hat{\mathrm{B}}^{(\lambda)} \in \underset{\mathrm{B} \in \mathbb{R}^{p \times q}}{\arg\min} \frac{1}{2} \|Y - X\mathrm{B}\|_F^2 + \lambda \|(\mathrm{B})\|_{2,1} := \mathcal{P}^{(\lambda)}(\mathrm{B}) \ , \tag{1}$$

with $\lambda > 0$ the regularization parameter controlling the trade-off between data fitting and regularization. The associated dual problem reads (see for instance [14])

$$\hat{\Theta}^{(\lambda)} = \underset{\Theta \in \Delta_X}{\arg\max} \tfrac{1}{2} \|Y\|_F^2 - \tfrac{\lambda^2}{2} \|\Theta - \tfrac{Y}{\lambda}\|_F^2 := \mathcal{D}^{(\lambda)}(\Theta) \ . \tag{2}$$

where $\Delta_X = \{\Theta \in \mathbb{R}^{n \times q} \ : \ \|X^\top \Theta\|_{2,\infty} \leq 1\}$ is the dual feasible set. The duality gap is defined by $\mathcal{G}^{(\lambda)}(\mathrm{B}, \Theta) := \mathcal{P}^{(\lambda)}(\mathrm{B}) - \mathcal{D}^{(\lambda)}(\Theta)$, for $\Theta \in \Delta_X$ [1].

2 From screening rules to working sets

The idea behind safe screening rules is to safely discard features from (1) as soon as it is guaranteed that the associated regression coefficients will be zero at convergence. The Gap Safe rules proposed in [14] for the multi-task regression read as follows: for a pair of primal-dual variables B and Θ, it is safe to discard feature j in the optimization problem (1) if:

$$\left\|X_{:,j}^\top \Theta\right\| + \|X_{:,j}\| \sqrt{\frac{2}{\lambda^2} \mathcal{G}^{(\lambda)}(\mathrm{B}, \Theta)} < 1 \Leftrightarrow d_j(\Theta) := \frac{1 - \left\|X_{:,j}^\top \Theta\right\|}{\|X_{:,j}\|} > \sqrt{\frac{2}{\lambda^2} \mathcal{G}^{(\lambda)}(\mathrm{B}, \Theta)} \ . \tag{3}$$

In other words, the duality gap value allows to define a threshold that is compared to $d_j(\Theta)$ in order to safely discard feature j. A natural idea to eliminate more features, while sacrificing safety, is to use the d_j's to prioritize features. One way is to introduce $r \in [0, 1]$ and only consider j if:

$$d_j(\Theta) \leq r \sqrt{\frac{2}{\lambda^2} \mathcal{G}^{(\lambda)}(\mathrm{B}, \Theta)} \ . \tag{4}$$

[1] When the dependency on X is needed, we write $\mathcal{P}^{(X,\lambda)}(\mathrm{B})$, for $\mathcal{P}^{(\lambda)}(\mathrm{B})$

This can be considered in an iterative strategy: starting from an initial value of B_0 (*e.g.,* $0 \in \mathbb{R}^{p\times q}$ or an approximate solution obtained for a close λ',one can obtain a feasible $\Theta_0 \in \Delta_X$. Given the primal-dual pair (B_0, Θ_0) one can compute d_j for all features and select the ones to be added to the working set $\mathcal{W}_1$. Then an *inner solver* can be started on $\mathcal{W}_1$. Assuming the inner solver returns a primal dual pair $(\tilde{\mathrm{B}}_t, \xi_t) \in \mathbb{R}^{p_t\times q} \times \mathbb{R}^{n\times q}$, where p_t is the size of $\mathcal{W}_t$, one can obtain a pair (B_t, ξ_t) by considering that $(\mathrm{B}_t)_{\mathcal{W}_t,:} = \tilde{\mathrm{B}}_t$ and $(\mathrm{B}_t)_{\bar{\mathcal{W}}_t,:} = 0$. Θ_t is then obtained from Θ_{t-1} and ξ_t as in [9]. We now to detail how to use d_j's to construct $\mathcal{W}_t$. A first strategy is to set a parameter r and then consider all features that satisfy (4). Yet this strategy does not offer a flexible control of the size of $\mathcal{W}_t$. A second strategy, which we use here, is to limit the number of features that shall enter $\mathcal{W}_t$. Constraining the size of $\mathcal{W}_t$ to be at most twice the size of $\mathcal{S}^r_{\mathrm{B}_{t-1}}$, we keep in $\mathcal{W}_t$ the blocks with indices in $\mathcal{S}^r_{\mathrm{B}_{t-1}}$ and add to it the ones in $\bar{\mathcal{S}}^r_{\mathrm{B}_{t-1}}$ with the smallest $d_j(\Theta_t)$. The iterative WS strategy is summarized in Algorithm 1. When combined with the BCD inner solver described in Section 3, we call it A5G (for AGGressive Gap, Greedy with Gram).

Algorithm 1: A5G

input: $X, Y, \lambda, p_0 = 100, \bar{\epsilon} = 10^{-6}, \underline{\epsilon} = 0.3$
Init : $\xi_0 = Y/\lambda, \Theta_0 = 0_{n,q}, \mathrm{B}_0 = 0_{p,q}$,
for $t = 1, \dots, T$ **do**
  $\alpha_t = \max\{\alpha \in [0,1] : (1-\alpha)\Theta_{t-1} + \alpha\xi_{t-1} \in \Delta_X\}$
  $\Theta_t = (1-\alpha_t)\Theta_{t-1} + \alpha_t\xi_{t-1}$
  `// global gap:`
  $g_t = \mathcal{G}^{(X,\lambda)}(\mathrm{B}_{t-1}, \Theta_t)$
  if $g_t \leq \bar{\epsilon}$ **then**
    Break
  for $j = 1, \dots, p$ **do**
    Compute $d_j^t = (1 - \|X_{:,j}^\top \Theta_t\|)/\|X_{:,j}\|$
    `// safe screening:`
    Remove j^{th} column of X if $d_j^t > \sqrt{2g_t/\lambda^2}$
  `// keep active features:`
  Set $(d^t)_{\mathcal{S}^r_{\mathrm{B}_{t-1}}} = -1$
  `// clipping:`
  $p_t = \max(p_0, \min(2\,\|\mathrm{B}_{t-1}\|_{2,0}, p))$
  $\mathcal{W}_t = \{j \in [p] : d_j^t$ among p_t lowest values of $d^t\}$
  `// Approximately solve sub-problem:`
  Get $\tilde{\mathrm{B}}_t, \xi_t \in \mathbb{R}^{p_t\times q} \times \Delta_{X_{:,\mathcal{W}_t}}$ s.t. $\mathcal{G}^{(X_{:,\mathcal{W}_t},\lambda)}(\tilde{\mathrm{B}}_t, \xi_t) \leq \underline{\epsilon} g_t$
  Set $\mathrm{B}_t \in \mathbb{R}^{p\times q}$ s.t. $(\mathrm{B}_t)_{\mathcal{W}_t,:} = \tilde{\mathrm{B}}_t$ $(\mathrm{B}_t)_{\bar{\mathcal{W}}_t,:} = 0$.
return B_t

3 Block Coordinate Descent (BCD) as inner solver

We now address the choice of the inner solver to minimize (1) once the WS has been defined. We minimize $\mathcal{P}^{(\lambda)}(\mathrm{B}) = f(\mathrm{B}) + \lambda\sum_{j=1}^p \|\mathrm{B}_j\|$, where $f(\mathrm{B}) = \|Y - X\mathrm{B}\|_F^2/2$. In this section, $\mathrm{B}_j \in \mathbb{R}^{1\times q}$ is the j^{th} row of B. In classical BCD algorithms, a block (line) j_k is chosen according to a particular selection rule, then updated with:$\mathrm{B}^k_{j_k} = \mathcal{T}_{j_k,L_{j_k}}(\mathrm{B}^{k-1}) = \mathrm{prox}_{\frac{\lambda}{L}\|\cdot\|}\left(\mathrm{B}_j - \frac{1}{L}\nabla_j f(\mathrm{B})\right)$, with $L_j = \|X_{:,j}\|^2$ and for $z \in \mathbb{R}^q, \mu > 0$, $\mathrm{prox}_{\mu\|\cdot\|}(z) = \arg\min_{x\in\mathbb{R}^q} \frac{1}{2}\|z-x\|^2 + \mu\|x\| = \mathrm{BST}(z,\mu) := \left(1 - \frac{\mu}{\|z\|}\right)_+ z$, where for any real number a, $(a)_+ = \max(0, a)$.

3.1 Greedy / Gauss-Southwell strategies

Following [17], we introduce a variant of the Gauss-Southwell (GS) rule. Contrary to static selection strategies such as the cyclic [3, 2] ($j_k = k \pmod p$) and the random one [16] (where j_k is drawn uniformly in $[p]$) these variants aim at identifying the "best" block to be updated. The GS-r variant picks the block maximizing the length of the update: $j_k \in \arg\max_{j\in[p]} \|\mathcal{T}_{j,L_j}(\mathrm{B}^{k-1}) - \mathrm{B}^{k-1}_j\|$. To reduce the cost of this rule, we use a variant, GS-rB, which looks for the best features only in batches of size B, chosen in a cyclic fashion (experiments are done with $B = 10$)

3.2 Gram matrix precomputation

As it selects the best block for each update, the GS-rB rule decreases the number of epochs needed to reach convergence. Yet, the heavier computation to pick the block can cancel this benefit. However, when the Gram matrix $Q = [Q_1, \dots, Q_p] = X^\top X$ is stored (which is possible since the subproblems are small), it becomes tractable to maintain the gradients $H^k = X^\top(X\mathrm{B}^k - Y) \in \mathbb{R}^{p\times q}$. The BCD steps becomes

$$\begin{cases} \delta\mathrm{B}_j & \leftarrow \mathrm{BST}\left(\mathrm{B}_j^{k-1} - \frac{1}{L_j}H_j^{k-1}, \frac{\lambda}{L_j}\right) - \mathrm{B}_j^{k-1} \\ \mathrm{B}_j^k & \leftarrow \mathrm{B}_j^{k-1} + \delta\mathrm{B}_j \quad \text{if} \quad \delta\mathrm{B}_j \neq 0 \\ H^k & \leftarrow H^{k-1} + Q_j\delta\mathrm{B}_j \quad \text{if} \quad \delta\mathrm{B}_j \neq 0 \end{cases} . \tag{5}$$

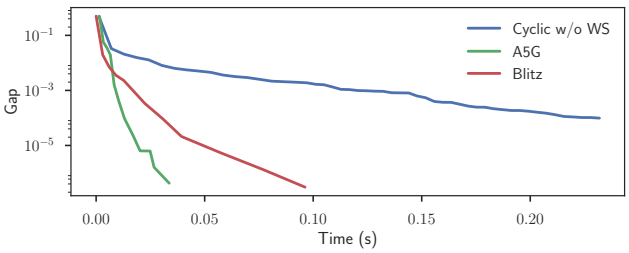

Figure 1: Duality gap as a function of time for the Lasso on the standard Leukemia dataset ($n = 72, p = 7129$) using $\lambda = 0.01\|X^\top Y\|_{2,\infty}$. Methods compared are the cyclic BCD from `scikit-learn` (Cyclic w/o WS), the C++ implementation of BLITZ as well as our WS approach with the GS-rB rule ($B = 10$) with precomputation of the Gram matrix. Both WS approaches outperform the plain BCD solver.

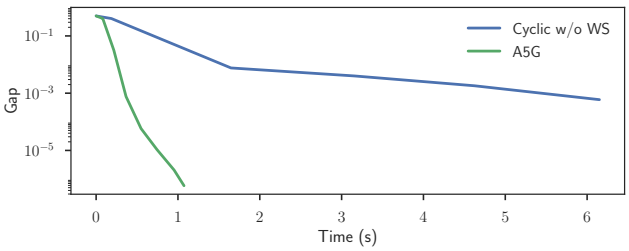

Figure 2: Duality gap as a function of time for the multi-task Lasso on MEG data ($n = 302, p = 7498, q = 181$) for $\lambda = 0.1\|X^\top Y\|_{2,\infty}$. The cyclic BCD from `scikit-learn` is compared to the WS approach with the GS-rB rule ($B = 10$) with precomputation of the Gram matrix. The proposed WS approach outperforms the plain BCD solver.

If the update is 0, the only computation required is the first line, which is $\mathcal{O}(q)$ since the gradients are stored. If the value of B_j changes, the additional costs are the update of B_j and a rank one update of the gradients. This low cost make the use of GS-rB rule possible to accelerate the subproblems resolution.

4 Experiments

First we consider the Lasso problem which allows us to compare our implementation to the state-of-the-art C++ implementation of BLITZ by [9]. We only compare to BLITZ, since extensive experiments in [9] indicated that it is currently the fastest solver for the Lasso. Figure 1 presents the duality gap as a function of time on the Leukemia dataset. Our implementation reaches comparable performance with the BLITZ C++ implementation, which is itself significantly better than the `scikit-learn` implementation [20] (no working set strategy) and faster than the GLMNET R Package according to [9].

Figure 2 presents results for multi-task Lasso problems, relevant to brain imaging with magneto- and electroencephalography (M/EEG) [30]. Y and B are multivariate time-series. Here, $n = 302$ corresponds to the number of sensors, $q = 181$ to the number of time instants and $p = 7498$ to the number of brain locations. The multi-task Lasso allows to identify brain activity stable on a short time interval [19]. In this experiment, we use data (from the MNE dataset, see [8]) following an auditory stimulation in the left ear, in fixed orientation setting. We set $\lambda = 0.1\lambda_{\max}$, which leads to 24 blocks with non-zero coefficients at convergence (*i.e.,* 24 active brain locations).

5 Conclusion and future work

We have proposed a connection between Gap Safe screening rules and working set (WS) strategies, such as BLITZ, to tackle many sparse learning problems, such as $\ell_{2,1}$ regularized regression. We have shown that in the context of small subproblems, precomputing the Gram matrix allows the Gauss-Southwell rule to reach comparable performance to cyclic updates, not only in terms of epochs but also in terms of computing time. To our knowledge, our implementation is the first to demonstrate timing performance for GS rules. In particular, a GS variant we coined GS-rB, relying on restricting the search of the best update to small batches of blocks has provided the best compromise. Among possible improvements, more refined batch GS strategies could be investigated. Additionally, improving the efficiency of the stopping criterion strategies would be another venue for future research. Finally, the impact of the growth of the WS size would benefit from further studies.

References

[1] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Convex optimization with sparsity-inducing norms. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[2] A. Beck, E. Pauwels, and S. Sabach. The cyclic block conditional gradient method for convex optimization problems. *SIAM J. Optim.*, 25(4):2024–2049, 2015.

[3] A. Beck and L. Tetruashvili. On the convergence of block coordinate type methods. *SIAM J. Imaging Sci.*, 23(4):651–694, 2013.

[4] P. J. Bickel, Y. Ritov, and A. B. Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *Ann. Statist.*, 37(4):1705–1732, 2009.

[5] L. El Ghaoui, V. Viallon, and T. Rabbani. Safe feature elimination in sparse supervised learning. *J. Pacific Optim.*, 8(4):667–698, 2012.

[6] O. Fercoq, A. Gramfort, and J. Salmon. Mind the duality gap: safer rules for the lasso. In *ICML*, pages 333–342, 2015.

[7] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *Ann. Appl. Stat.*, 1(2):302–332, 2007.

[8] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Lauri Parkkonen, and Matti S. Hämäläinen. MNE software for processing MEG and EEG data. *NeuroImage*, 86:446 – 460, Feb 2014.

[9] T. B. Johnson and C. Guestrin. Blitz: A principled meta-algorithm for scaling sparse optimization. In *ICML*, pages 1171–1179, 2015.

[10] T. B. Johnson and C. Guestrin. Unified methods for exploiting piecewise linear structure in convex optimization. In *NIPS*, pages 4754–4762, 2016.

[11] K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *J. Mach. Learn. Res.*, 8(8):1519–1555, 2007.

[12] M. Kowalski, P. Weiss, A. Gramfort, and S. Anthoine. Accelerating ISTA with an active set strategy. In *OPT 2011: 4th International Workshop on Optimization for Machine Learning*, page 7, 2011.

[13] M. Loth. *Active Set Algorithms for the LASSO*. PhD thesis, Université des Sciences et Technologie de Lille - Lille I, 2011.

[14] E. Ndiaye, O. Fercoq, A. Gramfort, and J. Salmon. Gap safe screening rules for sparse multi-task and multi-class models. In *NIPS*, pages 811–819, 2015.

[15] E. Ndiaye, O. Fercoq, A. Gramfort, and J. Salmon. Gap safe screening rules for sparsity enforcing penalties. Technical report, 2016.

[16] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optim.*, 22(2):341–362, 2012.

[17] J. Nutini, M. W. Schmidt, I. H. Laradji, M. P. Friedlander, and H. A. Koepke. Coordinate descent converges faster with the Gauss-Southwell rule than random selection. In *ICML*, pages 1632–1641, 2015.

[18] M. R. Osborne, B. Presnell, and B. A. Turlach. A new approach to variable selection in least squares problems. *IMA J. Numer. Anal.*, 20(3):389–403, 2000.

[19] W. Ou, M. Hämaläinen, and P. Golland. A distributed spatio-temporal EEG/MEG inverse solver. *NeuroImage*, 44(3):932–946, Feb 2009.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

[21] Z. Peng, T. Wu, Y. Xu, M. Yan, and W. Yin. Coordinate friendly structures, algorithms and applications. *arXiv preprint arXiv:1601.00863*, 2016.

[22] V. Roth and B. Fischer. The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *ICML*, pages 848–855, 2008.

[23] S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Math. Program.*, 155(1):105–145, 2016.

[24] S. K. Shevade and S. S. Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19(17):2246–2253, 2003.

[25] H.-J. M. Shi, S. Tu, Y. Xu, and W. Yin. A primer on coordinate descent algorithms. *arXiv preprint arXiv:1610.00040*, 2016.

[26] R. V. Southwell. Relaxation methods in engineering science - a treatise on approximate computation. *The Mathematical Gazette*, 25(265):180–182, 1941.

[27] R. Tibshirani, J. Bien, J. Friedman, T. Hastie, N. Simon, J. Taylor, and R. J. Tibshirani. Strong rules for discarding predictors in lasso-type problems. *J. Roy. Statist. Soc. Ser. B*, 74(2):245–266, 2012.

[28] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.*, 109(3):475–494, 2001.

[29] P. Tseng and S. Yun. Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *J. Optim. Theory Appl.*, 140(3):513, 2009.

[30] D. P. Wipf, J. P. Owen, H. Attias, K. Sekihara, and S. S. Nagarajan. Estimating the location and orientation of complex, correlated neural activity using MEG. In *NIPS*, pages 1777–1784. 2008.

[31] T. T. Wu and K. Lange. Coordinate descent algorithms for lasso penalized regression. *Ann. Appl. Stat.*, pages 224–244, 2008.