Node-screening tests for the ℓ_0-penalized least-squares problem

Paper ID : 2482

Théo Guyard*, Cédric Herzet†, Clément Elvira‡

*Applied Mathematics Department, INSA Rennes, France | †SIMSMART team, INRIA Rennes-Bretagne Atlantique, France | ‡SCEE team, CentraleSupelec Rennes, France

Objectives

Reduce the optimization time of a Branch-and-Bound (BnB) solving the ℓ_0-penalized least-squares problem by detecting nodes of the search tree that cannot yield a global optimizer.

Introduction

Finding a sparse representation is a fundamental problem in the field of statistics, machine learning and inverse problems, among others. It consists in decomposing some input vector $\mathbf{y} \in \mathbf{R}^m$ as a linear combination of a few columns of a dictionary $\mathbf{A} \in \mathbf{R}^{m \times n}$. This task can be addressed by solving

$$p^\star = \begin{cases} \min & \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_0 \\ \text{s.t.} & \|\mathbf{x}\|_\infty \leq M \end{cases} \qquad (P)$$

where $\|\mathbf{x}\|_0$ counts the number of non-zeros in $\mathbf{x}$, $\lambda > 0$ is a tuning parameter and M is a large constant. Interestingly, (P) can be reformulated as a Mixed-Integer Program by introducing binary variables encoding the nullity of $\mathbf{x}$ entries. It has been shown to be solvable for moderate-size problems by commercial solvers or tailored BnB algorithms.

Atamtürk and Gómez recently extended the notion of *screening* introduced by El Ghaoui *et al.* from convex sparse problems to ℓ_0-penalized problems. Their methodology allows to *detect the positions of some zero and non-zero entries* in the minimizers of a problem similar to (P) so as to reduce the problem dimensionality in preprocessing.

We make one step forward in the development of numerical methods addressing large-scale ℓ_0-penalized problems by proposing *node-screening* rules allowing to *prune nodes* within the BnB search tree. Moreover, we emphasize the existence of a *nesting property* between our node-screening tests at different nodes. This enables to *prune multiple nodes* at any step of the optimization process with a marginal cost.

BnB procedures

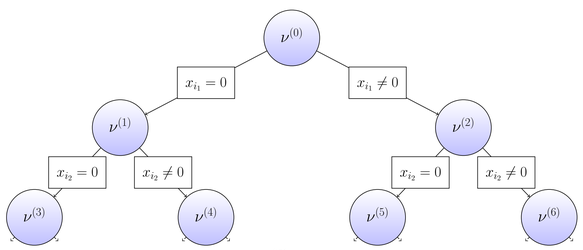
Particularized to problem (P), a BnB can be interpreted as a *search tree* where a new decision regarding the nullity of an entry of the variable $\mathbf{x}$ is taken at each node. A node of the tree corresponds to a triplet $\nu = (\mathcal{S}_0, \mathcal{S}_1, \bar{\mathcal{S}})$ where $\mathcal{S}_0$ and $\mathcal{S}_1$ contain the indices of $\mathbf{x}$ which are forced to be zero and non-zero and where $\bar{\mathcal{S}}$ gathers all the unfixed indices of $\mathbf{x}$. Starting from a root node with $\mathcal{S}_0 = \mathcal{S}_1 = \emptyset$, the BnB alternates between processing the current node and selecting a new node. Its efficiency depends on the *number of nodes* treated and on the ability to *process them efficiently*.

When processing node $\nu = (\mathcal{S}_0, \mathcal{S}_1, \bar{\mathcal{S}})$, we prospect if any $\mathbf{x}$ with zeros on $\mathcal{S}_0$ and non-zeros on $\mathcal{S}_1$ can yield a global minimizer of (P). To that end, we impose these constraints on (P) and solve

$$p_l^\nu = \begin{cases} \min & \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \frac{\lambda}{M}\|\mathbf{x}_{\bar{\mathcal{S}}}\|_1 + \lambda|\mathcal{S}_1| \\ \text{s.t.} & \|\mathbf{x}\|_\infty \leq M,\ \mathbf{x}_{\mathcal{S}_0} = \mathbf{0} \end{cases} \qquad (P_l^\nu)$$

which is a *relaxation* of the problem obtained. The above problem is a constrained LASSO and can be solved in polynomial time.

Then, we compare p_l^ν to the best known upper bound p_u on $p^\star$. It $p_l^\nu > p_u$, then node ν cannot yield a global optimizer of (P) and can therefore be pruned. The bound p_u is obtained by constructing feasible solutions to (P) at each node. If the node is not pruned, the tree exploration goes on. When all nodes have been either explored or pruned, the BnB algorithm stops and any candidate yielding the best upper bound p_u is a minimizer of (P).



(a) BnB search tree. Node $\nu^{(0)}$ is the root node. Other nodes correspond to different choices of $\mathcal{S}_0$ and $\mathcal{S}_1$.

Node-screening tests

Our node-screening methodology aims at identifying nodes of the BnB tree that cannot yield a global optimizer of (P). In particular, we leverage properties of the *Fenchel dual* of (P_l^ν) given by

$$\max \left\{ \mathrm{D}_l^\nu(\mathbf{u}) = \tfrac{1}{2}\|\mathbf{y}\|_2^2 - \tfrac{1}{2}\|\mathbf{y} - \mathbf{u}\|_2^2 - \sum_{i \in \bar{\mathcal{S}}} [\boldsymbol{\gamma}_i(\mathbf{u})]_+ - \sum_{i \in \mathcal{S}_1} \boldsymbol{\gamma}_i(\mathbf{u}) \right\} \qquad (D_l^\nu)$$

with $[w]_+ = \max(w, 0)$ and $\boldsymbol{\gamma}_i(\mathbf{u}) = M|\mathbf{A}_i^{\mathrm{T}}\mathbf{u}| - \lambda$.

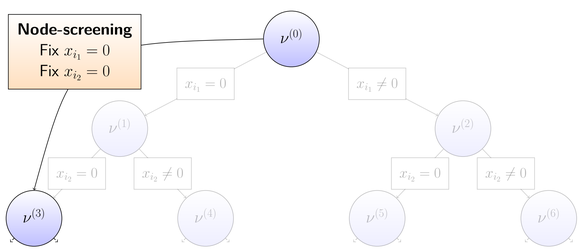
Note that the objective of (D_l^ν) is composed of a term common to all nodes and terms corresponding to the current fixed variables. Thus, it only differs from one term between two consecutive nodes. A direct consequence is the following result.

Node-screening tests

At node $\nu = (\mathcal{S}_0, \mathcal{S}_1, \bar{\mathcal{S}})$, $\forall i \in \bar{\mathcal{S}}$ and $\forall \mathbf{u}$,

- If $\mathrm{D}_l^\nu(\mathbf{u}) + [\boldsymbol{\gamma}_i(\mathbf{u})]_+ > p_u$, then *index i can be safely swapped from $\bar{\mathcal{S}}$ to $\mathcal{S}_1$*.
- If $\mathrm{D}_l^\nu(\mathbf{u}) + [-\boldsymbol{\gamma}_i(\mathbf{u})]_+ > p_u$, then *index i can be safely swapped from $\bar{\mathcal{S}}$ to $\mathcal{S}_0$*.

Stated otherwise, node-screening tests potentially allow to fix new variables at node ν, *i.e.,* to prune nodes of the BnB tree without having to process them. Furthermore, they can be implemented in a very efficient way *within* most solution method tailored to (P_l^ν). Hence, our methodology allows to *prune nodes without any additional complexity*.



(b) Impact of node-screening on the BnB. Node-screening tests are passed at $\nu^{(0)}$, allowing to discard a part of the tree.

Numerical results

To assess our methodology, we compare three different algorithmic solutions to address (P) :

- A direct solution method using CPLEX
- A tailored BnB algorithm
- A tailored BnB algorithm with node-screening

To generate problem data, we consider the two following setups : 1) The elements of $\mathbf{A}$ are i.i.d. realizations of gaussian distribution. 2) $\mathbf{A}$ has a Toeplitz structure with shifted versions of a sinc curve. We set (m, n) to $(500, 100)$ for Gaussian dictionaries and to $(500, 300)$ for Toeplitz dictionaries. Then, we sample a k-sparse vector $\mathbf{x}^0$ with $k \in \{5, 7, 9\}$ and we set $\mathbf{y} = \mathbf{A}\mathbf{x}^0 + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon}$ corresponds to a 10dB noise. The values of λ and M are tuned statistically to recover $\mathbf{x}^0$ when solving (P). Results are averaged over 100 instances.

		Direct			BnB			BnB+scr		
	k	N	T	F	N	T	F	N	T	F
Gaussian	5	96	25.9	0	70	1.5	0	**56**	**0.7**	**0**
	7	292	60.8	0	180	5.1	0	**152**	**3.0**	**0**
	9	781	102.6	10	483	15.6	0	**412**	**9.8**	**0**
Toeplitz	5	1,424	10.2	0	965	6.4	0	**725**	**4.2**	**0**
	7	17,647	106.5	0	10,461	79.3	0	**7,881**	**52.2**	**0**
	9	80,694	353.4	50	47,828	346.4	48	**41,166**	**267.0**	**40**

Table: Number of nodes explored (N), solving time in sec (T) and number of instances not solved within 10^3 sec (F).

One observes that `BnB+scr` outperforms the two other methods on all scenarii. We also note that, compared to `BnB`, the *reduction in the optimization time* is more significant than the *reduction in the number of nodes* processed. A thorough examination of our results indicates that the processing step is performed all the faster as many variables are fixed to zero in (P_l^ν). Our node-screening methodology allows to reach quickly nodes where the bounding step is performed with a lower computational cost. The overall solving time is therefore even more reduced.