Logic Rules: Sparse Learning and Chordal Graphs

Alper Atamtürk[1], **Anna Deza**[1], Andrés Gómez[2]

[1]Department of Industrial Engineering and Operations Research,
University of California, Berkeley

[2]Department of Industrial and Systems Engineering,
University of Southern California

ISMP (Montréal)

July 25, 2024

Table of Contents

Sparsity in Decision-Making Models

Learning model makes predictions based on a feature vector with p features, $x \in \mathbb{R}^p$. Suppose our model takes the form:

$$\text{prediction}_i = f\left(\beta^\top x_i\right),$$

where $\beta \in \mathbb{R}^p$ are the regression coefficients.

Sparsity in Decision-Making Models

Learning model makes predictions based on a feature vector with p features, $x \in \mathbb{R}^p$. Suppose our model takes the form:

$$\text{prediction}_i = f\left(\beta^\top x_i\right),$$

where $\beta \in \mathbb{R}^p$ are the regression coefficients.

Often, **sparsity** is desired: can be cheaper to deploy, better out-of-sample performance, and important for interpretability.

Sparsity in Decision-Making Models

Learning model makes predictions based on a feature vector with p features, $x \in \mathbb{R}^p$. Suppose our model takes the form:

$$\text{prediction}_i = f\left(\beta^\top x_i\right),$$

where $\beta \in \mathbb{R}^p$ are the regression coefficients.

Often, **sparsity** is desired: can be cheaper to deploy, better out-of-sample performance, and important for interpretability.

Measure sparsity with ℓ_0 pseudo-norm: $\|\beta\|_0 = \sum_{i=1}^p \mathrm{I}(\beta_i \neq 0)$.

How do we build Sparse Machine Learning Models?

Given n observations with p features, $(x_i \in \mathbb{R}^p, y_i \in \mathbb{R})_{i=1}^n$, we train models by optimizing model coefficients $\beta \in \mathbb{R}^p$ that minimize loss function $\mathcal{L}(\beta)$.

$$\textbf{(REG)} \quad \zeta_R = \min_{\beta \in \mathbb{R}^p} \mathcal{L}(\beta) + \underbrace{\lambda \|\beta\|_2^2}_{\text{Shrinkage}} + \underbrace{\mu \|\beta\|_0}_{\text{Sparsity}}.$$

How do we build Sparse Machine Learning Models?

Given n observations with p features, $(x_i \in \mathbb{R}^p, y_i \in \mathbb{R})_{i=1}^n$, we train models by optimizing model coefficients $\beta \in \mathbb{R}^p$ that minimize loss function $\mathcal{L}(\beta)$.

$$\textbf{(REG)} \quad \zeta_R = \min_{\beta \in \mathbb{R}^p} \mathcal{L}(\beta) + \underbrace{\lambda \|\beta\|_2^2}_{\text{Shrinkage}} + \underbrace{\mu \|\beta\|_0}_{\text{Sparsity}}.$$

$$\textbf{(CARD)} \quad \zeta_C = \min_{\beta \in \mathbb{R}^p} \mathcal{L}(\beta) + \underbrace{\lambda \|\beta\|_2^2}_{\text{Shrinkage}} \;\text{ s.t }\; \underbrace{\|\beta\|_0 \leq k}_{\text{Sparsity}}$$

How do we build Sparse Machine Learning Models?

Given n observations with p features, $(x_i \in \mathbb{R}^p, y_i \in \mathbb{R})_{i=1}^n$, we train models by optimizing model coefficients $\beta \in \mathbb{R}^p$ that minimize loss function $\mathcal{L}(\beta)$.

$$\textbf{(REG)} \quad \zeta_R = \min_{\beta \in \mathbb{R}^p} \mathcal{L}(\beta) + \underbrace{\lambda \|\beta\|_2^2}_{\text{Shrinkage}} + \underbrace{\mu \|\beta\|_0}_{\text{Sparsity}}.$$

$$\textbf{(CARD)} \quad \zeta_C = \min_{\beta \in \mathbb{R}^p} \mathcal{L}(\beta) + \underbrace{\lambda \|\beta\|_2^2}_{\text{Shrinkage}} \ \text{s.t} \ \underbrace{\|\beta\|_0 \leq k}_{\text{Sparsity}}$$

- Exact sparsity makes this **NP-hard**
- Recent interest in using mixed integer optimization techniques to solve sparse learning to optimality

MIO for Sparse Learning

Goal: formulate sparse learning as a mixed integer optimization problem.

$$\zeta_R = \min_{\beta} \mathcal{L}(\beta) + \lambda \|\beta\|_2^2 + \mu \|\beta\|_0$$

MIO for Sparse Learning

1 Introduce **binary indicator** variables z to model ℓ_0: $z_i = 0 \Rightarrow \beta_i = 0$.

$$
\begin{aligned}
\zeta_R = \min_{\beta, z} \; & \mathcal{L}(\beta) + \lambda \sum_{i=1}^{p} \beta_i^2 + \mu \sum_{i=1}^{p} z_i \\
\text{s.t.} \; & \beta_i (1 - z_i) = 0, \quad j \in [p] \\
& \beta \in \mathbb{R}^p, \; z \in \{0, 1\}^p
\end{aligned}
$$

MIO for Sparse Learning

2 Reformulate using the **perspective** function for each β_i^2:

$$\zeta_R = \min_{\beta \in \mathbb{R}^p,\ z \in \{0,1\}^p} \mathcal{L}(\beta) + \lambda \sum_{i=1}^{p} \frac{\beta_i^2}{z_i} + \mu \sum_{i=1}^{p} z_i$$

$\rightarrow$ can be represented with conic quadratic constraints

MIO for Sparse Learning

- Have a MIP formulation, but scalability is a concern
- Atamtürk and Gómez (2020) introduce $\ell_2 - \ell_0$ **screening rules**: Use solution to the convex relaxation to **safely** prune or fix features ($z_i = 0$ or $z_i = 1$)
  - $\rightarrow$ Eliminate features *guaranteed* to not be in the optimal solution
  - $\rightarrow$ **Reduce the dimension** of problem before the full optimization step, *improving solution times*

Table of Contents

Atamtürk and Gómez (2020): Screening Rules

Derived based on the dual of the perspective terms β_i^2/z_i, and only require solving the convex relaxation of the sparse learning problem.

Atamtürk and Gómez (2020): Screening Rules

Sparse learning problem:

$$\zeta_R = \min_{\beta \in \mathbb{R}^p,\ z \in \{0,1\}^p} \mathcal{L}(\beta) + \lambda \sum_{i=1}^{p} \frac{\beta_i^2}{z_i} + \mu \sum_{i=1}^{p} z_i$$

Atamtürk and Gómez (2020): Screening Rules

Relax binary constraints & take **Fenchel dual** of the perspective function:

$$\zeta(w) = \min_{\substack{\beta\in\mathbb{R}^p,\\ z\in[0,1]^p}} \mathcal{L}(\beta) + \sum_{i=1}^{p} \lambda w_i \beta_i + \sum_{i=1}^{p} \left(\mu - \lambda \frac{w_i^2}{4} \right) z_i$$

$\rightarrow \zeta(w)$ gives a *lower bound* for ζ_R for any dual $w \in \mathbb{R}^p$

Atamtürk and Gómez (2020): Screening Rules

Relax binary constraints & take **Fenchel dual** of the perspective function:

$$\zeta(w) = \min_{\substack{\beta \in \mathbb{R}^p, \\ z \in [0,1]^p}} \mathcal{L}(\beta) + \sum_{i=1}^{p} \lambda w_i \beta_i + \sum_{i=1}^{p} \left(\mu - \lambda \frac{w_i^2}{4} \right) z_i$$

$\rightarrow \zeta(w)$ gives a *lower bound* for ζ_R for any dual $w \in \mathbb{R}^p$

$\mu - \lambda \frac{w_i^2}{4}$ determines z_i

Atamtürk and Gómez (2020): Screening Rules

Relax binary constraints & take **Fenchel dual** of the perspective function:

$$\zeta(w) = \min_{\substack{\beta \in \mathbb{R}^p, \\ z \in [0,1]^p}} \mathcal{L}(\beta) + \sum_{i=1}^{p} \lambda w_i \beta_i + \sum_{i=1}^{p} \left(\mu - \lambda \frac{w_i^2}{4} \right) z_i$$

$\rightarrow \zeta(w)$ gives a *lower bound* for ζ_R for any dual $w \in \mathbb{R}^p$

$\mu - \lambda \frac{w_i^2}{4} > 0 \Rightarrow z_i = 0$

Atamtürk and Gómez (2020): Screening Rules

Relax binary constraints & take **Fenchel dual** of the perspective function:

$$\zeta(w) = \min_{\substack{\beta \in \mathbb{R}^p, \\ z \in [0,1]^p}} \mathcal{L}(\beta) + \sum_{i=1}^{p} \lambda w_i \beta_i + \sum_{i=1}^{p} \left(\mu - \lambda \frac{w_i^2}{4} \right) z_i$$

$\rightarrow \zeta(w)$ gives a *lower bound* for ζ_R for any dual $w \in \mathbb{R}^p$

$\mu - \lambda \frac{w_i^2}{4} > 0 \Rightarrow z_i = 0$

Lower bound: $\zeta(w) + \mu - \lambda \frac{w_i^2}{4} \leq \zeta_R(z_i = 1)$

Atamtürk and Gómez (2020): Screening Rules

Relax binary constraints & take **Fenchel dual** of the perspective function:

$$\zeta(w) = \min_{\substack{\beta \in \mathbb{R}^p, \\ z \in [0,1]^p}} \mathcal{L}(\beta) + \sum_{i=1}^{p} \lambda w_i \beta_i + \sum_{i=1}^{p} \left(\mu - \lambda \frac{w_i^2}{4} \right) z_i$$

$\rightarrow \zeta(w)$ gives a *lower bound* for ζ_R for any dual $w \in \mathbb{R}^p$

$\mu - \lambda \frac{w_i^2}{4} > 0 \Rightarrow z_i = 0$

Lower bound: $\zeta(w) + \mu - \lambda \frac{w_i^2}{4} \leq \zeta_R(z_i = 1)$

$\zeta(w) + \mu - \lambda \frac{w_i^2}{4} > \underbrace{\zeta_u}_{\text{upper bound}} \Rightarrow$ no optimal solution has $z_i = 1$.

Atamtürk and Gómez (2020): Screening Rules

Prop. (Atamtürk and Gómez (2020)) *Safe Screening for* **(REG)**

For any dual variable $w \in \mathbb{R}^p$, let $\alpha = \mu - \lambda \frac{w^2}{4}$ and ζ_u an upper bound on **(REG)**. Then any optimal solution z^* to **(REG)** satisfies the following rule given the corresponding condition holds.

Condition	**Screening Rule**
$\zeta(w) + \alpha_i > \zeta_u$	$z_i^* = 0$
$\zeta(w) - \alpha_i > \zeta_u$	$z_i^* = 1$

Very effective if relaxation gap is small, but degrades as gap increases.

Atamtürk and Gómez (2020): Screening Rules

Prop. (Atamtürk and Gómez (2020)) *Safe Screening for* **(REG)**

For any dual variable $w \in \mathbb{R}^p$, let $\alpha = \mu - \lambda \frac{w^2}{4}$ and ζ_u an upper bound on **(REG)**. Then any optimal solution z^* to **(REG)** satisfies the following rule given the corresponding condition holds.

Condition	**Screening Rule**
$\zeta(w) + \alpha_i > \zeta_u$	$z_i^* = 0$
$\zeta(w) - \alpha_i > \zeta_u$	$z_i^* = 1$

Very effective if relaxation gap is small, but degrades as gap increases.

$\rightarrow$ We introduce logic rules, generalizing screening rules: consider the **logical relationships** between **groups** of features

Table of Contents

Introducing Logic Rules

Logic rules screen for:

Simultaneous **inclusion** of features	Simultaneous **exclusion** of features	Pairwise **ranking** of features
$z_i + z_j \leq 1$	$z_i + z_j \geq 1$	$z_i \leq z_j$

Screening rules derivation: If $\zeta(w) + \alpha_i > \zeta_u \Rightarrow \mathbf{z_i^*} = \mathbf{0}$.

Logic rules: want a lower bound for $\zeta_R(z_i = 1, z_j = 1)$

Introducing Logic Rules

Logic rules screen for:

Simultaneous **inclusion** of features	Simultaneous **exclusion** of features	Pairwise **ranking** of features
$z_i + z_j \leq 1$	$z_i + z_j \geq 1$	$z_i \leq z_j$

Screening rules derivation: If $\zeta(w) + \alpha_i > \zeta_u \Rightarrow \mathbf{z_i^*} = \mathbf{0}$.

Logic rules: want a lower bound for $\zeta_R(z_i = 1, z_j = 1)$

$\rightarrow$ Natural lower bound: $\zeta(w) + \alpha_i + \alpha_j$

$\rightarrow$ $\zeta(w) + \alpha_i + \alpha_j > \zeta_u \Rightarrow \mathbf{z_j} + \mathbf{z_k} \leq \mathbf{1}$

Logic Rules

Prop. Logic Rules for **(REG)**

For any dual variable $w \in \mathbb{R}^p$, let $\alpha_i = \mu - \frac{\lambda}{4} w_i^2$ and ζ_u be an upper bound on **(REG)**. Then any optimal solution z^* to **(REG)** satisfies the following rule given the corresponding condition holds.

Condition	**Logic Rule**
$\zeta(w) - \alpha_j - \alpha_k > \zeta_u$	$z_i + z_j \geq 1$
$\zeta(w) + \alpha_i + \alpha_j > \zeta_u$	$z_i + z_j \leq 1$
$\zeta(w) + \alpha_i - \alpha_j > \zeta_u$	$z_i - z_j \leq 0$
$\zeta(w) - \alpha_i + \alpha_j > \zeta_u$	$z_i - z_j \leq 0$

Logic Rules

Prop. Logic Rules for **(REG)**

For any dual variable $w \in \mathbb{R}^p$, let $\alpha_i = \mu - \frac{\lambda}{4} w_i^2$ and ζ_u be an upper bound on **(REG)**. Then any optimal solution z^* to **(REG)** satisfies the following rule given the corresponding condition holds.

Condition	**Logic Rule**
$\zeta(w) - \alpha_j - \alpha_k > \zeta_u$	$z_i + z_j \geq 1$
$\zeta(w) + \alpha_i + \alpha_j > \zeta_u$	$z_i + z_j \leq 1$
$\zeta(w) + \alpha_i - \alpha_j > \zeta_u$	$z_i - z_j \leq 0$
$\zeta(w) - \alpha_i + \alpha_j > \zeta_u$	$z_i - z_j \leq 0$

- On their own, pairwise interactions do not help much

  $\rightarrow$ We want to construct stronger relationships, and leverage them

How can logic rules help?

Solvers can exploit constraints of the type $\sum_{i \in S} z_i \leq 1$ (**SOS1**)

- More effective branching
- Better preprocessing, heuristics, cutting planes, may lead to order-of-magnitudes speedup (Fischer & Pfetsch, 2018)

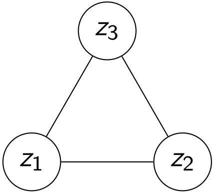
How can logic rules help?

Solvers can exploit constraints of the type $\sum_{i \in S} z_i \leq 1$ (**SOS1**)

- More effective branching
- Better preprocessing, heuristics, cutting planes, may lead to order-of-magnitudes speedup (Fischer & Pfetsch, 2018)

We can use logic rules to get such implications.

$$\begin{aligned} z_1 + z_2 &\leq 1 \\ z_1 + z_3 &\leq 1 \\ z_2 + z_3 &\leq 1 \\ z &\in \{0,1\}^3 \end{aligned} \quad \Rightarrow \quad \begin{aligned} z_1 + z_2 + z_3 &\leq 1 \\ z &\in \{0,1\}^3 \end{aligned}$$



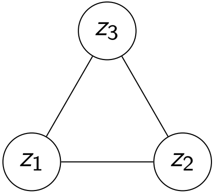
How can logic rules help?

Solvers can exploit constraints of the type $\sum_{i \in S} z_i \leq 1$ (**SOS1**)

- More effective branching
- Better preprocessing, heuristics, cutting planes, may lead to order-of-magnitudes speedup (Fischer & Pfetsch, 2018)

We can use logic rules to get such implications.

$$\begin{aligned} z_1 + z_2 &\leq 1 \\ z_1 + z_3 &\leq 1 \\ z_2 + z_3 &\leq 1 \\ z &\in \{0,1\}^3 \end{aligned} \quad \Rightarrow \quad \begin{aligned} z_1 + z_2 + z_3 &\leq 1 \\ z &\in \{0,1\}^3 \end{aligned}$$



Build **conflict graph** to get clique inequalities

Conflict Graph

Logic rules imply a conflict graph $G = (V, E)$ with $V = \{1, \ldots, p\}$ and $E = \{(i, j) : z_i + z_j \leq 1, i, j \in V\}$.

Conflict Graph

Logic rules imply a conflict graph $G = (V, E)$ with $V = \{1, \ldots, p\}$ and $E = \{(i, j) : z_i + z_j \leq 1, i, j \in V\}$.

For any clique C, at most one vertex can be selected: $\sum_{i \in C} z_i \leq 1$.

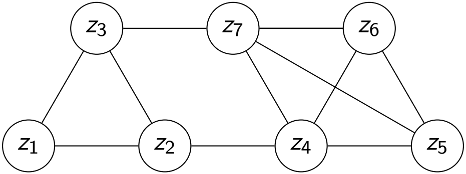
Example:

$z_1 + z_2 \leq 1, z_1 + z_2 \leq 1, z_2 + z_3 \leq 1$

$z_4 + z_5 \leq 1, z_4 + z_6 \leq 1, z_5 + z_7 \leq 1$

$z_5 + z_6 \leq 1, z_5 + x_7 \leq 1, z_6 + z_7 \leq 1$

$z_2 + z_4 \leq 1, z_3 + z_7 \leq 1$



Conflict Graph

Logic rules imply a conflict graph $G = (V, E)$ with $V = \{1, \ldots, p\}$ and $E = \{(i, j) : z_i + z_j \leq 1, i, j \in V\}$.

For any clique C, at most one vertex can be selected: $\sum_{i \in C} z_i \leq 1$.

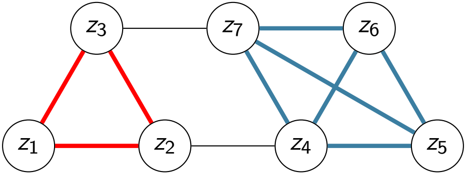
Example:

$z_1 + z_2 \leq 1, z_1 + z_2 \leq 1, z_2 + z_3 \leq 1$

$z_4 + z_5 \leq 1, z_4 + z_6 \leq 1, z_5 + z_7 \leq 1$

$z_5 + z_6 \leq 1, z_5 + x_7 \leq 1, z_6 + z_7 \leq 1$

$z_2 + z_4 \leq 1, z_3 + z_7 \leq 1$



Conflict Graph

Logic rules imply a conflict graph $G = (V, E)$ with $V = \{1, \ldots, p\}$ and $E = \{(i, j) : z_i + z_j \leq 1, i, j \in V\}$.

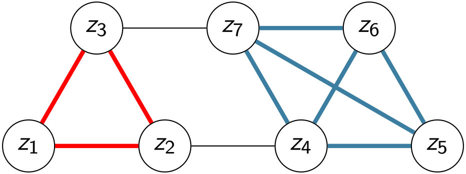
For any clique C, at most one vertex can be selected: $\sum_{i \in C} z_i \leq 1$.

Example:

$z_1 + z_2 + z_3 \leq 1$

$z_4 + z_5 + z_6 + z_6 \leq 1$

$z_2 + z_4 \leq 1, z_3 + z_7 \leq 1$



- To make strongest implications, want to identify **all maximal cliques**

Conflict Graph

Logic rules imply a conflict graph $G = (V, E)$ with $V = \{1, \ldots, p\}$ and $E = \{(i, j) : z_i + z_j \leq 1, i, j \in V\}$.

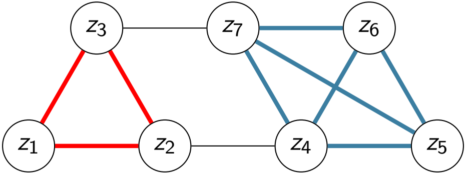
For any clique C, at most one vertex can be selected: $\sum_{i \in C} z_i \leq 1$.

Example:

$z_1 + z_2 + z_3 \leq 1$

$z_4 + z_5 + z_6 + z_6 \leq 1$

$z_2 + z_4 \leq 1, z_3 + z_7 \leq 1$



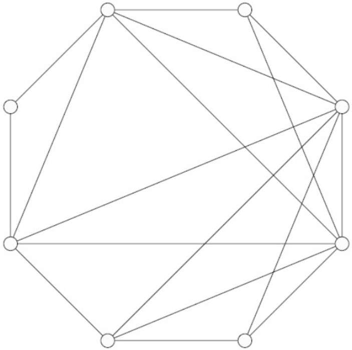
- To make strongest implications, want to identify **all maximal cliques**
- For general graphs: exponentially many maximal cliques...

$\rightarrow$ We can exploit special structure of G generated by logic rules

Chordal Graphs

Definition (Chordal Graph)

A graph is chordal if every cycle of length at least four contains a chord, that is, an edge between two non-adjacent vertices on the cycle.



Chordal Graphs

Theorem

A chordal graph has linearly many maximal cliques.

Chordal Graphs

Theorem

A chordal graph has linearly many maximal cliques.

The conflict graph G generated by the logic rules is chordal.

$\rightarrow$ Can exploit structure to find all maximal cliques in polynomial time

Finding Maximal Cliques in Logic Rules Conflict Graph

Prop. *Polynomial Time Algorithm for Maximal Cliques*

We propose a $\mathcal{O}(p \log p)$ time algorithm to find **all maximal cliques** generated by the conflict graph implied by the logic rules.

Finding Maximal Cliques in Logic Rules Conflict Graph

Prop. *Polynomial Time Algorithm for Maximal Cliques*

We propose a $\mathcal{O}(p \log p)$ time algorithm to find **all maximal cliques** generated by the conflict graph implied by the logic rules.

Key: Logic Rules say $\alpha_i + \alpha_j > \zeta_u - \zeta(w) \Rightarrow (i, j) \in E$.

Finding Maximal Cliques in Logic Rules Conflict Graph

Prop. *Polynomial Time Algorithm for Maximal Cliques*

We propose a $\mathcal{O}(p \log p)$ time algorithm to find **all maximal cliques** generated by the conflict graph implied by the logic rules.

Key: Logic Rules say $\alpha_i + \alpha_j > \zeta_u - \zeta(w) \Rightarrow (i, j) \in E$.

Suppose α happens to be sorted in decreasing order.

- If $\alpha_i + \alpha_{i+n} > \zeta_u - \zeta(w)$

Finding Maximal Cliques in Logic Rules Conflict Graph

Prop. *Polynomial Time Algorithm for Maximal Cliques*

We propose a $\mathcal{O}(p \log p)$ time algorithm to find **all maximal cliques** generated by the conflict graph implied by the logic rules.

Key: Logic Rules say $\alpha_i + \alpha_j > \zeta_u - \zeta(w) \Rightarrow (i, j) \in E$.

Suppose α happens to be sorted in decreasing order.

- If $\alpha_i + \alpha_{i+n} > \zeta_u - \zeta(w) \Rightarrow \alpha_\ell + \alpha_{i+n} > \zeta_u - \zeta(w)\ \forall \ell = 1, \ldots, i$.
- Then we have $\{1, 2, \ldots, i, i+n\}$ is a clique.

Finding Maximal Cliques in Logic Rules Conflict Graph

Prop. *Polynomial Time Algorithm for Maximal Cliques*

We propose a $\mathcal{O}(p \log p)$ time algorithm to find **all maximal cliques** generated by the conflict graph implied by the logic rules.

Key: Logic Rules say $\alpha_i + \alpha_j > \zeta_u - \zeta(w) \Rightarrow (i, j) \in E$.

Suppose α happens to be sorted in decreasing order.

- If $\alpha_i + \alpha_{i+n} > \zeta_u - \zeta(w) \Rightarrow \alpha_\ell + \alpha_{i+n} > \zeta_u - \zeta(w)\ \forall \ell = 1, \dots, i$.
- Then we have $\{1, 2, \dots, i, i+n\}$ is a clique.

Algorithm: sort α, then scan. Notice: no need to build G!

Overview of logic rules

Stage 1: Make building blocks.
Find logical relationships between *pairs* of features, equivalent to exclusivity constraints.

$\rightarrow$ Solve a convex problem

Stage 2: Construct stronger inequalities.
Construct stronger inequalities implied by Stage 1, equivalent to finding maximal cliques in a conflict graph.

$\rightarrow$ Exploit special structure to efficiently find all maximal cliques

Overview of logic rules

Stage 1: Make building blocks.
Find logical relationships between *pairs* of features, equivalent to exclusivity constraints.

$\rightarrow$ Solve a convex problem

Stage 2: Construct stronger inequalities.
Construct stronger inequalities implied by Stage 1, equivalent to finding maximal cliques in a conflict graph.

$\rightarrow$ Exploit special structure to efficiently find all maximal cliques

Output: $\{C_1, C_2, \ldots, C_t\}$ such that $\sum_{i \in C_j} z_i \leq 1$ is safe to add to our sparse learning problem.

Table of Contents

Numerical Results

We solve (cardinality-constrained) sparse linear regression on synthetic and real data sets and compare:

- Gurobi alone
- Gurobi + screening rules
- Gurobi + screening rules + logic rules

Synthetic Data

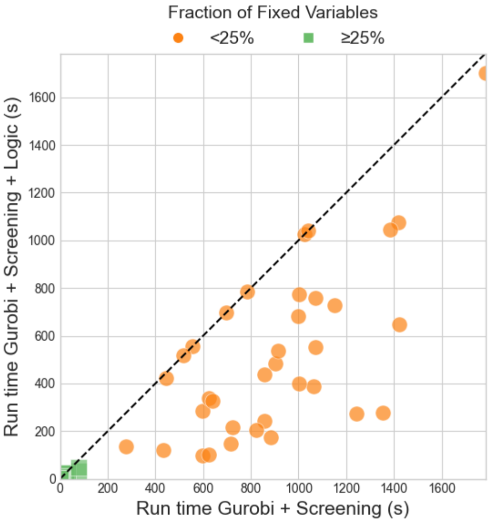
We generate synthetic data with 1000 features, 100 observations, and vary the noise level (SNR) and ℓ_2 regularization strength (λ). Solve with cardinality constraint, $k = 10$.

Synthetic Data

We generate synthetic data with 1000 features, 100 observations, and vary the noise level (SNR) and ℓ_2 regularization strength (λ). Solve with cardinality constraint, $k = 10$.

SNR	λ	**RGap %**	**GRB Runtime (s)**	**GRB + Screen Runtime (s)**	**GRB + Screen + Logic Runtime (s)**
0.05	1/10	47.4	1,572	1,574	981
	1/8	31.6	1,083	1,083	581
	1/4	7.5	761	4.1	3.8
	1/2	1.6	439	0.7	0.7
1.0	1/10	39.8	728	733	482
	1/8	28.5	715	505	266
	1/4	7.5	646	7.8	6.4
	1/2	1.5	386	0.6	0.6
6.0	1/10	25.5	684	276	57
	1/8	20.3	755	163	40
	1/4	6.0	605	1.0	1.0
	1/2	1.4	527	0.5	0.5
Average		**18.2**	**741**	**362**	**202**

Synthetic Data



Real Data Results

We use the Riboflavin dataset, $4,088$ with 71 observations, varying the sparsity constraint (k) and ℓ_2 regularization strength (λ). We solve with a one-hour time limit, reporting the end gap if no solution is found.

Real Data Results

We use the Riboflavin dataset, $4,088$ with 71 observations, varying the sparsity constraint (k) and ℓ_2 regularization strength (λ). We solve with a one-hour time limit, reporting the end gap if no solution is found.

k/p	$10^2\lambda_0$	**GRB + Screen Runtime (s)**	**GRB + Screen + Logic Runtime (s)**
0.25%	2.5	(35%)	(3%)
	4	91	87
	5	39	38
0.5%	2.5	128	109
	4	47	42
	5	38	38
1.25%	2.5	187	156
	4	50	47
	5	37	37

Real Data Results

We use the Riboflavin dataset, $4,088$ with 71 observations, varying the sparsity constraint (k) and ℓ_2 regularization strength (λ). We solve with a one-hour time limit, reporting the end gap if no solution is found.

k/p	$10^2\lambda_0$	GRB + Screen Runtime (s)	GRB + Screen + Logic Runtime (s)
0.25%	2.5	(35%)	(3%)
	4	91	87
	5	39	38
0.5%	2.5	128	109
	4	47	42
	5	38	38
1.25%	2.5	187	156
	4	50	47
	5	37	37

k/p	$10^2\lambda_0$	GRB + Screen Runtime (s)	GRB + Screen + Logic Runtime (s)
2.5%	2.5	(66%)	(24%)
	4	(59%)	(11%)
	5	1,607	3,044
3.5%	2.5	(46%)	(14%)
	4	1,134	622
	5	1,441	399
5%	2.5	2,677	1,290
	4	1,228	434
	5	3,101	410

Conclusion

Logic rules are general preprocessing framework that generates inequalities that can be leveraged by mixed integer optimization solvers to speed up sparse learning computation.

- Proposed method is **efficient** due to the exploitation of the underlying structure (chordality) in the conflict graph generated by the inequalities.
- Complements screening rules by helping in cases where they are unsuccessful (large relaxation gaps).