Scaling up the LASSO with interaction features

Marine Le Morvan
Joint work with **Jean-Philippe Vert**

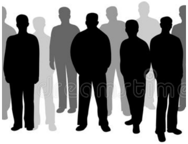
CBIO - Mines Paristech, INSERM U900 - Curie institute, Paris, France

November 7^{th}, 2017

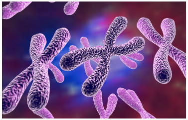
Motivating example

DNA sequences

P_1 . . . A T C G C T G A A T A C G G C T C G A A A T C G G A . . . ✓
P_2 . . . T T C G G T G A G T A C G G C T C G A A A T C G G A . . . ✗
P_3 . . . A T C G C T G A A T A C G G C T C G A A A T C G G A . . . ✗
P_4 . . . T T C G C T G A G T A C G G C T C G A A A T C G G A . . . ✓
P_5 . . . T T C G C T G A G T A C G G C T C G A A A T C G G A . . . ✓
P_6 . . . A T C G G T G A G T A C G G T T C G A T A T C G G A . . . ✗
P_7 . . . A T C G G T G A A T A C G G T T C G A T A T C G G A . . . ✗
P_8 . . . T T C G G T G A G T A C G G C T C G A T A T C G G A . . . ✓



Sequence $\longrightarrow$



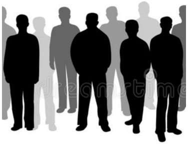
Query $\longrightarrow$

Response to treatment?
Disease risk?
Drug assimilation rate?
Ancestry?
. . .

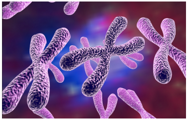
Motivating example

DNA sequences

P_1	...	A	T	C	G	C	T	G	A	A	T	A	C	G	G	C	T	C	G	A	A	A	T	C	G	G	A	...	✓
P_2	...	T	T	C	G	G	T	G	A	G	T	A	C	G	G	C	T	C	G	A	A	A	T	C	G	G	A	...	✗
P_3	...	A	T	C	G	C	T	G	A	A	T	A	C	G	G	C	T	C	G	A	A	A	T	C	G	G	A	...	✗
P_4	...	T	T	C	G	C	T	G	A	G	T	A	C	G	G	C	T	C	G	A	A	A	T	C	G	G	A	...	✓
P_5	...	T	T	C	G	C	T	G	A	G	T	A	C	G	G	C	T	C	G	A	A	A	T	C	G	G	A	...	✓
P_6	...	A	T	C	G	G	T	G	A	G	T	A	C	G	G	T	T	C	G	A	T	A	T	C	G	G	A	...	✗
P_7	...	A	T	C	G	G	T	G	A	A	T	A	C	G	G	T	T	C	G	A	T	A	T	C	G	G	A	...	✗
P_8	...	T	T	C	G	G	T	G	A	G	T	A	C	G	G	C	T	C	G	A	T	A	T	C	G	G	A	...	✓
		↑				↑				↑						↑					↑								



Sequence $\longrightarrow$



Query $\longrightarrow$

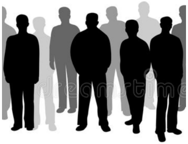
Response to treatment?
Disease risk?
Drug assimilation rate?
Ancestry?
...

Motivating example

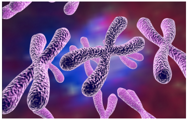
DNA sequences

		s_1	—	—	—	s_2	—	—	—	s_3	—	—	—	—	—	s_4	—	—	—	—	s_5	—	—	—	—	—	—		y
P_1	...	A	T	C	G	C	T	G	A	A	T	A	C	G	G	C	T	C	G	A	A	A	T	C	G	G	A	...	✓
P_2	...	T	T	C	G	G	T	G	A	G	T	A	C	G	G	C	T	C	G	A	A	A	T	C	G	G	A	...	✗
P_3	...	A	T	C	G	C	T	G	A	A	T	A	C	G	G	C	T	C	G	A	A	A	T	C	G	G	A	...	✗
P_4	...	T	T	C	G	C	T	G	A	G	T	A	C	G	G	C	T	C	G	A	A	A	T	C	G	G	A	...	✓
P_5	...	T	T	C	G	C	T	G	A	G	T	A	C	G	G	C	T	C	G	A	A	A	T	C	G	G	A	...	✓
P_6	...	A	T	C	G	G	T	G	A	G	T	A	C	G	G	T	T	C	G	A	T	A	T	C	G	G	A	...	✗
P_7	...	A	T	C	G	G	T	G	A	A	T	A	C	G	G	T	T	C	G	A	T	A	T	C	G	G	A	...	✗
P_8	...	T	T	C	G	G	T	G	A	G	T	A	C	G	G	C	T	C	G	A	T	A	T	C	G	G	A	...	✓
		↑				↑				↑						↑					↑								

Single Nucleotide Polyphormisms (SNPs)



Sequence ⟶



Query ⟶

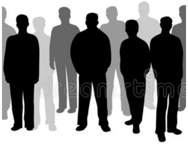
Response to treatment?
Disease risk?
Drug assimilation rate?
Ancestry?
...

Motivating example

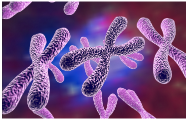
DNA sequences

		s_1	s_2	s_3	s_4	s_5		y
P_1	...	1	0	1	0	0	...	✓
P_2	...	0	1	0	0	0	...	✗
P_3	...	1	0	1	0	0	...	✗
P_4	...	0	0	0	0	0	...	✓
P_5	...	0	0	0	0	0	...	✓
P_6	...	1	1	0	1	1	...	✗
P_7	...	1	1	1	1	1	...	✗
P_8	...	0	1	0	0	1	...	✓
		↑	↑	↑	↑	↑		

Single Nucleotide Polyphormisms (SNPs)



Sequence ⟶



Query ⟶

Response to treatment?
Disease risk?
Drug assimilation rate?
Ancestry?
...

Motivating example

The LASSO is commonly used to predict $\boldsymbol{y}$:

$$\underbrace{\begin{pmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_n \end{pmatrix}}_{\in \mathbb{R}^n} \approx \underbrace{\begin{pmatrix} | & | & & | \\ \boldsymbol{X}_1 & \boldsymbol{X}_2 & \dots & \boldsymbol{X}_n \\ | & | & & | \end{pmatrix}}_{\boldsymbol{X} \in [\![0,1]\!]^{n \times p}} \cdot \underbrace{\begin{pmatrix} \boldsymbol{w}_1^* \\ \vdots \\ \boldsymbol{w}_n^* \end{pmatrix}}_{\in \mathbb{R}^p}$$

Typically, $n << p$.

$$\boldsymbol{w}^* \leftarrow \underset{\boldsymbol{w} \in \mathbb{R}^p}{\operatorname{argmin}} \ \frac{1}{n} \underbrace{\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2}_{\text{data fitting term}} + \underbrace{\lambda \|\boldsymbol{w}\|_1}_{\text{sparsity inducing penalty}} \qquad \text{(LASSO)}$$

The penalty forces only a few features to be selected in the model, for ex:

$$\boldsymbol{y} = \boldsymbol{w}_1^* \boldsymbol{X}_{s_1} + \boldsymbol{w}_4^* \boldsymbol{X}_{s4} + \boldsymbol{w}_5^* \boldsymbol{X}_{s_5}$$

Motivating example

We would like to also consider second order interaction effects of the form:

$$\boldsymbol{X}_j \odot \boldsymbol{X}_k, \quad (j, k) \in [\![1, p]\!]^2$$

where $\odot$ is the Hadamard product (=entrywise product).

Typically, we would like to be able to learn a model such as:

$$\boldsymbol{y} = \boldsymbol{w}_1^* \boldsymbol{X}_{s_1} + \boldsymbol{w}_4^* \boldsymbol{X}_{s_4} + \boldsymbol{w}_5^* \boldsymbol{X}_{s_5} + \boldsymbol{w}_{1,2}^* \boldsymbol{X}_{s_1} \odot \boldsymbol{X}_{s_2}$$

Motivating example

We would like to also consider second order interaction effects of the form:

$$\boldsymbol{X}_j \odot \boldsymbol{X}_k, \quad (j,k) \in [\![1,p]\!]^2$$

where $\odot$ is the Hadamard product (=entrywise product).

Typically, we would like to be able to learn a model such as:

$$\boldsymbol{y} = \boldsymbol{w}_1^* \boldsymbol{X}_{s_1} + \boldsymbol{w}_4^* \boldsymbol{X}_{s_4} + \boldsymbol{w}_5^* \boldsymbol{X}_{s_5} + \boldsymbol{w}_{1,2}^* \boldsymbol{X}_{s_1} \odot \boldsymbol{X}_{s_2}$$

Why is it interesting?

brown hair genes
AND
*MCR*1*-variant 1* $\Longrightarrow$

brown hair genes
AND
*MCR*1*-variant 2* $\Longrightarrow$

We would like to also consider second order interaction effects of the form:

$$\boldsymbol{X}_j \odot \boldsymbol{X}_k, \quad (j,k) \in [\![1,p]\!]^2$$

where $\odot$ is the Hadamard product (=entrywise product).

Typically, we would like to be able to learn a model such as:

$$\boldsymbol{y} = \boldsymbol{w}_1^* \boldsymbol{X}_{s_1} + \boldsymbol{w}_4^* \boldsymbol{X}_{s_4} + \boldsymbol{w}_5^* \boldsymbol{X}_{s_5} + \boldsymbol{w}_{1,2}^* \boldsymbol{X}_{s_1} \odot \boldsymbol{X}_{s_2}$$

Why is it difficult?

The number of interactions terms is equal to:

$$D = \frac{p(p-1)}{2}$$

If $p = 100.000$, then $D = 5 \times 10^9$.
Classical LASSO solvers will be too slow.

This work aims at providing a framework to fit sparse linear models with second order interaction terms when the data is binary.

Problem formulation

$$\boldsymbol{y} = \underbrace{\begin{pmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_n \end{pmatrix}}_{\in \mathbb{R}^n}, \quad \boldsymbol{X} = \underbrace{\begin{pmatrix} | & | & & | \\ \boldsymbol{X}_1 & \boldsymbol{X}_2 & \dots & \boldsymbol{X}_n \\ | & | & & | \end{pmatrix}}_{\in [\![0,1]\!]^{n \times p}}, \quad \boldsymbol{Z} = \underbrace{\begin{pmatrix} | & | & & | & | & & | \\ \boldsymbol{X}_1 & \boldsymbol{X}_2 & \dots & \boldsymbol{X}_n & \boldsymbol{X}_1\boldsymbol{X}_n & \dots & \boldsymbol{X}_n\boldsymbol{X}_n \\ | & | & & | & | & & | \end{pmatrix}}_{\in [\![0,1]\!]^{n \times D}}$$

- We will indifferently use $\boldsymbol{X}_j \odot \boldsymbol{X}_k$ and $\boldsymbol{X}_j\boldsymbol{X}_k$.
- Primal problem

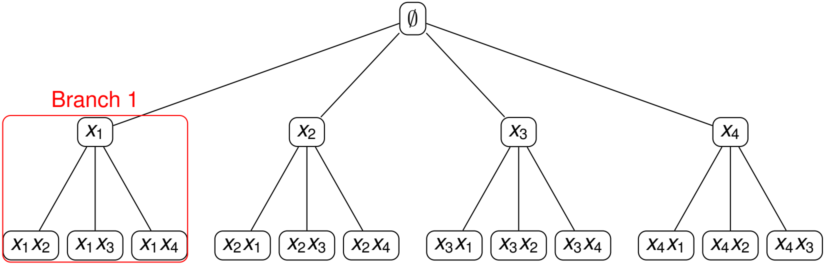
$$\min_{(\boldsymbol{w},b) \in \mathbb{R}^D \times \mathbb{R}} g_\lambda(\boldsymbol{w}, b) = \frac{1}{n} \|\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{w} - b\|_2^2 + \lambda \|\boldsymbol{w}\|_1 \tag{1}$$

- Dual problem

$$\max_{\boldsymbol{\theta} \in \mathbb{R}^n} f_\lambda(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{y}\|_2^2 - \frac{1}{2}\|\boldsymbol{\theta} - \boldsymbol{y}\|_2^2 \text{ s.t. } \begin{cases} \left|(\boldsymbol{X}_j\boldsymbol{X}_k)^T\boldsymbol{\theta}\right| \leq \lambda & (j,k) \in [\![1,p]\!]^2 \\ \boldsymbol{1}^T\boldsymbol{\theta} = 0 \end{cases} \tag{2}$$

Previous work

- Safe Pattern Pruning (SPP) (Nakagawa et al., 2016)
- SPP relies on **safe screening rules**. Given primal and dual feasible solutions, safe screening rules identify features which are guaranteed not be active at the optimum.
- The idea of SPP is to leverage **the tree structure of interactions features**, and propose a screening rule applicable to entire branches.



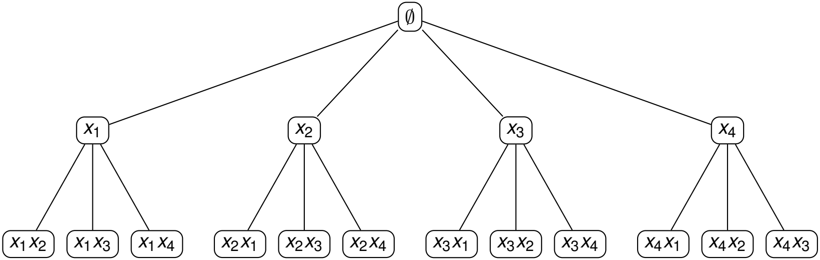
WHInter

- Limitations:
  - ✓ SPP does not allow to prune enough branches, especially when n increases.
  - ✓ The size of the safe set can be big (for medium values of λ)
  - ✓ A dual feasible point is expensive to compute.
- We propose **WHInter**:
  - ✓ Working set strategy.
  - ✓ New branch pruning strategy for the identification of the active set.
  - ✓ Efficient computation of branch bounds using a Maximum Inner Product Search (MIPS) framework for binary data.

WHInter achieves a **speed ups of up to one order of magnitude** compared to SPP.

WHInter pseudo algorithm

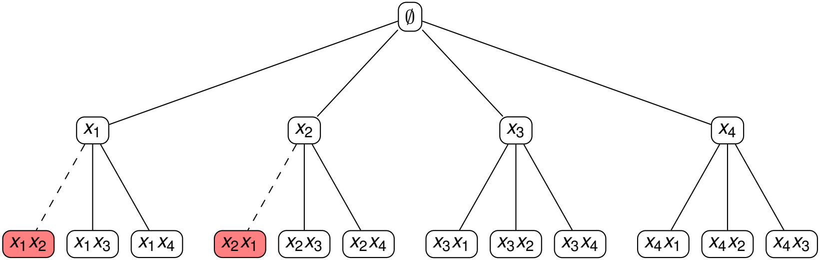
Input: $\boldsymbol{X} \in [\![0, 1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$$\mathcal{M}_\lambda = \{ \quad \}$$

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0,1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$

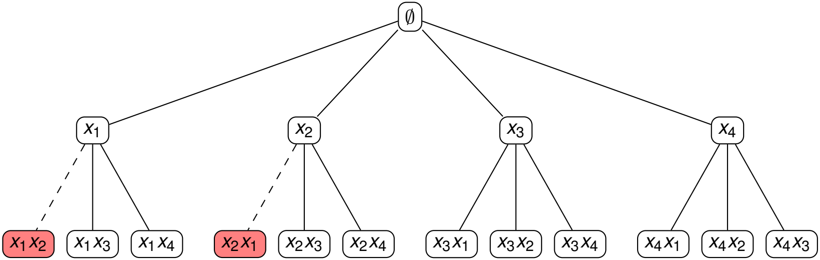


$$\mathcal{M}_\lambda = \left\{ \boxed{x_1 x_2} \right\}$$

Initialize $\mathcal{M}_\lambda$.

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0,1]\!]^{n\times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$\mathcal{M}_\lambda = \{ x_1 x_2 \}$

Initialize $\mathcal{M}_\lambda$.
Pre-solve and initialize ϕ.

$$\boldsymbol{w}, b \leftarrow \underset{(\boldsymbol{w},b)\in\mathbb{R}^D\times\mathbb{R}}{\operatorname{argmin}} \frac{1}{n}\|\boldsymbol{y} - \boldsymbol{Z}_{\mathcal{M}_\lambda}\boldsymbol{w} - b\|_2^2 + \lambda\|\boldsymbol{w}\|_1$$

$$\phi \leftarrow \boldsymbol{y} - \boldsymbol{Z}_{\mathcal{M}_\lambda}\boldsymbol{w} - b$$

Problem formulation

$$\boldsymbol{y} = \underbrace{\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}}_{\in \mathbb{R}^n}, \quad \boldsymbol{X} = \underbrace{\begin{pmatrix} | & | & & | \\ \boldsymbol{X}_1 & \boldsymbol{X}_2 & \dots & \boldsymbol{X}_n \\ | & | & & | \end{pmatrix}}_{\in [\![0,1]\!]^{n \times p}}, \quad \boldsymbol{Z} = \underbrace{\begin{pmatrix} | & | & & | & | & & | \\ \boldsymbol{X}_1 & \boldsymbol{X}_2 & \dots & \boldsymbol{X}_n & \boldsymbol{X}_1\boldsymbol{X}_n & \dots & \boldsymbol{X}_n\boldsymbol{X}_n \\ | & | & & | & | & & | \end{pmatrix}}_{\in [\![0,1]\!]^{n \times D}}$$

- The KKT conditions state that:

$$\forall (j,k) \in [\![1,p]\!]^2, \quad \left|(\boldsymbol{X}_j\boldsymbol{X}_k)^T\boldsymbol{\theta}^*\right| \in \begin{cases} \{\lambda\} & \text{if } \boldsymbol{w}^*_{j,k} \neq 0 \\ [-\lambda,\lambda] & \text{if } \boldsymbol{w}^*_{j,k} = 0 \end{cases} \tag{3}$$

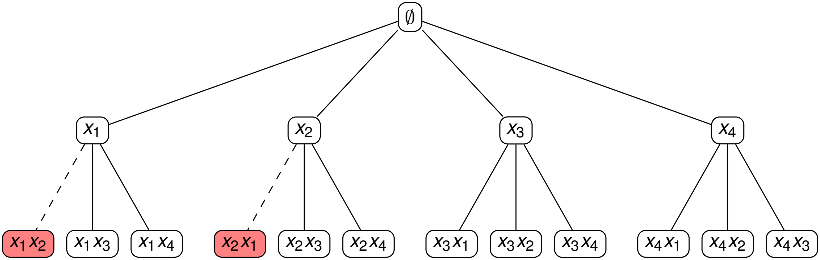
We will say that the constraint relative to $\boldsymbol{X}_j\boldsymbol{X}_k$ is violated whenever $\left|(\boldsymbol{X}_j\boldsymbol{X}_k)^T\boldsymbol{\theta}\right| > \lambda$

- The primal and dual optimal variables $(\boldsymbol{w}*, b*)$ and $\boldsymbol{\theta}^*$ are related as follows:

$$\boldsymbol{\theta}^* = \boldsymbol{y} - \boldsymbol{Z}\boldsymbol{w}^* - b^*$$

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0, 1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$$\mathcal{M}_\lambda = \left\{ \boxed{x_1 x_2} \right\}$$

 # Identify violated constraints and update working set.

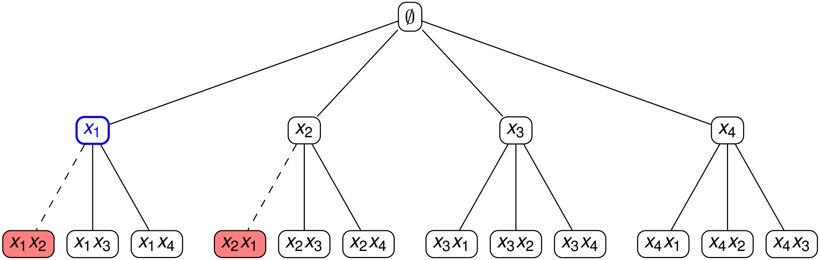
Compute the bound $\eta(\phi, \boldsymbol{X}_j)$.

We define $\eta(\phi, \boldsymbol{X}_j)$ as an upper bound on $\max_{\boldsymbol{X}_k :\, \boldsymbol{X}_j \boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| \left(\boldsymbol{X}_j \boldsymbol{X}_k\right)^T \phi \right|$.

If $\eta(\phi, \boldsymbol{X}_j) \leq \lambda$, then we know that all features in the branch respect the optimality conditions.

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0, 1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$\mathcal{M}_\lambda = \left\{ \boxed{x_1 x_2} \right\}$

 # Identify violated constraints and update working set.

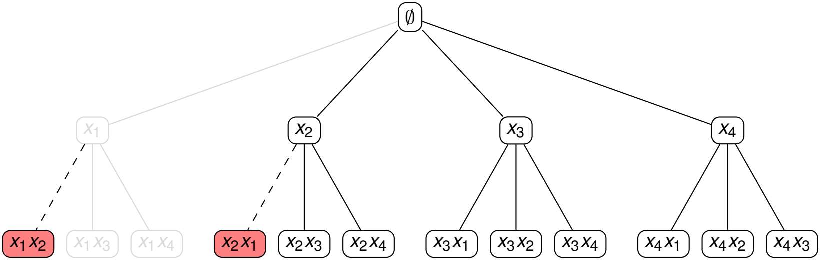
Compute the bound $\eta(\phi, \boldsymbol{X}_j)$.

We define $\eta(\phi, \boldsymbol{X}_j)$ as an upper bound on $\max\limits_{\boldsymbol{X}_k :\, \boldsymbol{X}_j \boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| \left(\boldsymbol{X}_j \boldsymbol{X}_k \right)^T \phi \right|$.

$$\eta(\phi, \boldsymbol{X}_1) < \lambda$$

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0,1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$\mathcal{M}_\lambda = \{ \boxed{x_1 x_2} \}$

 # Identify violated constraints and update working set.

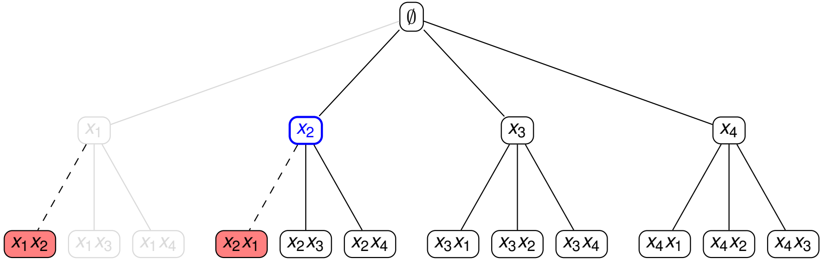
Compute the bound $\eta(\phi, \boldsymbol{X}_j)$.

We define $\eta(\phi, \boldsymbol{X}_j)$ as an upper bound on $\max\limits_{\boldsymbol{X}_k:\, \boldsymbol{X}_j \boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| (\boldsymbol{X}_j \boldsymbol{X}_k)^T \phi \right|$.

$$\eta(\phi, \boldsymbol{X}_1) < \lambda$$

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0, 1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$\mathcal{M}_\lambda = \{ x_1 x_2 \}$

 # Identify violated constraints and update working set.

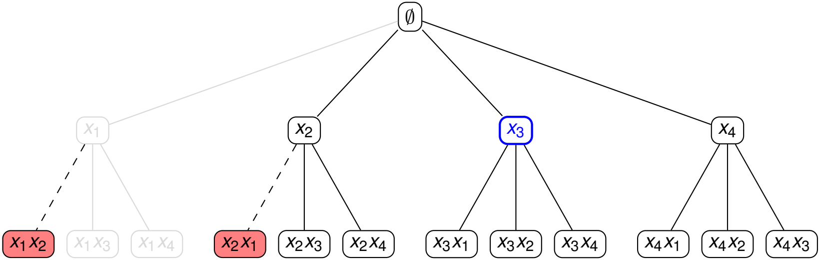
Compute the bound $\eta(\phi, \boldsymbol{X}_j)$.

We define $\eta(\phi, \boldsymbol{X}_j)$ as an upper bound on $\max_{\boldsymbol{X}_k :\, \boldsymbol{X}_j \boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| (\boldsymbol{X}_j \boldsymbol{X}_k)^T \phi \right|$.

$$\eta(\phi, \boldsymbol{X}_2) \geq \lambda$$

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0,1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$\mathcal{M}_\lambda = \{ \; x_1x_2 \; \}$

 # Identify violated constraints and update working set.

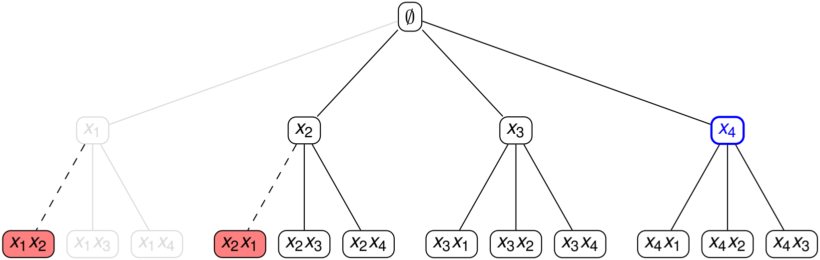
Compute the bound $\eta(\phi, \boldsymbol{X}_j)$.

We define $\eta(\phi, \boldsymbol{X}_j)$ as an upper bound on $\max_{\boldsymbol{X}_k :\, \boldsymbol{X}_j\boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| \left(\boldsymbol{X}_j\boldsymbol{X}_k\right)^T \phi \right|$.

$$\eta(\phi, \boldsymbol{X}_3) \geq \lambda$$

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0, 1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$\mathcal{M}_\lambda = \{ x_1 x_2 \}$

 # Identify violated constraints and update working set.

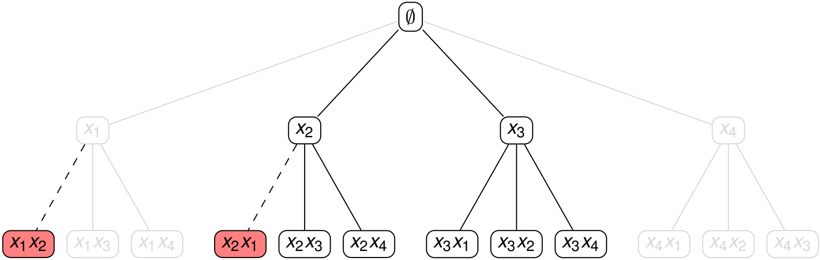
Compute the bound $\eta(\phi, \boldsymbol{X}_j)$.

We define $\eta(\phi, \boldsymbol{X}_j)$ as an upper bound on $\max_{\boldsymbol{X}_k :\, \boldsymbol{X}_j \boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| \left(\boldsymbol{X}_j \boldsymbol{X}_k \right)^T \phi \right|$.

$$\eta(\phi, \boldsymbol{X}_4) < \lambda$$

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0,1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$$\mathcal{M}_\lambda = \left\{ \boxed{X_1 X_2} \right\}$$

 # Identify violated constraints and update working set.

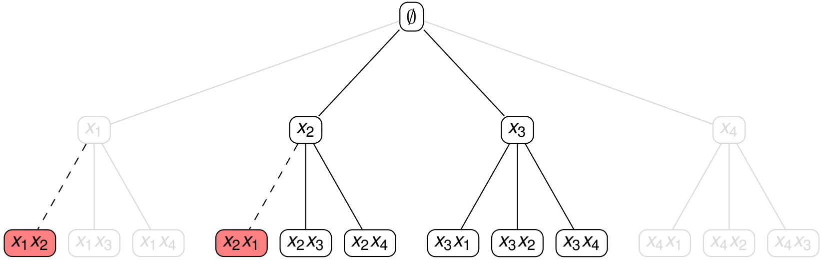
Compute the bound $\eta(\phi, \boldsymbol{X}_j)$.

We define $\eta(\phi, \boldsymbol{X}_j)$ as an upper bound on $\max\limits_{\boldsymbol{X}_k:\, \boldsymbol{X}_j \boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| \left(\boldsymbol{X}_j \boldsymbol{X}_k \right)^T \phi \right|$.

$$\eta(\phi, \boldsymbol{X}_4) < \lambda$$

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0,1]\!]^{n\times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$\mathcal{M}_\lambda = \{ \; x_1 x_2 \; \}$

 # Identify violated constraints and update working set.

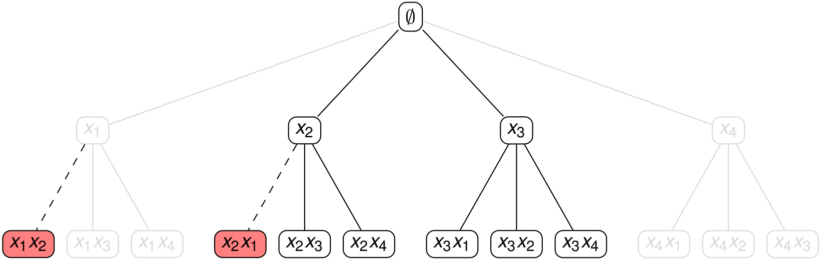
Compute the bound $\eta(\phi, \boldsymbol{X}_j)$.

We define $\eta(\phi, \boldsymbol{X}_j)$ as an upper bound on $\max\limits_{\boldsymbol{X}_k :\, \boldsymbol{X}_j\boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| \left(\boldsymbol{X}_j \boldsymbol{X}_k\right)^T \phi \right|$.

done

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0,1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



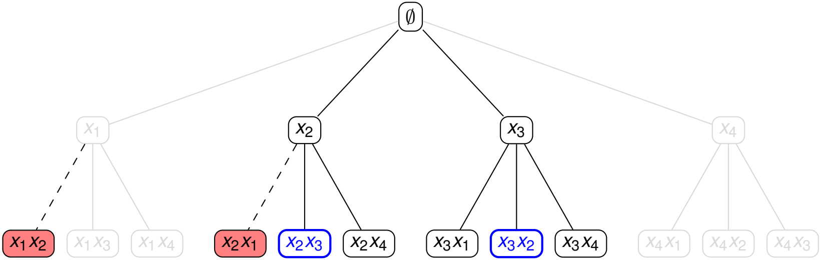
$\mathcal{M}_\lambda = \{ x_1 x_2 \}$

 # Identify violated constraints and update working set.
If $\eta(\phi, \boldsymbol{X}_j) \geq \lambda$, compute $\boldsymbol{m}_j(\phi)$ and update $\mathcal{M}_\lambda$.

$$\boldsymbol{m}_j(\phi) = \max_{\boldsymbol{X}_k :\, \boldsymbol{X}_j \boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| (\boldsymbol{X}_j \boldsymbol{X}_k)^T \phi \right|, \quad \mathcal{M}_\lambda \leftarrow \mathcal{M}_\lambda \cup \left\{ \boldsymbol{X}_j \boldsymbol{X}_k : \left| (\boldsymbol{X}_j \boldsymbol{X}_k)^t \phi \right| \geq \lambda \right\}$$

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0,1]\!]^{n\times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$\mathcal{M}_\lambda = \{ x_1 x_2 \}$

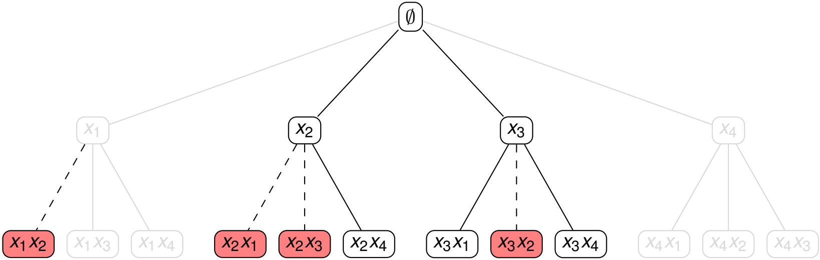
 # Identify violated constraints and update working set.
If $\eta(\phi, \boldsymbol{X}_j) \geq \lambda$, compute $\boldsymbol{m}_j(\phi)$ and update $\mathcal{M}_\lambda$.

$$\boldsymbol{m}_j(\phi) = \max_{\boldsymbol{X}_k:\, \boldsymbol{X}_j\boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| (\boldsymbol{X}_j\boldsymbol{X}_k)^T \phi \right|, \quad \mathcal{M}_\lambda \leftarrow \mathcal{M}_\lambda \cup \left\{ \boldsymbol{X}_j\boldsymbol{X}_k : \left| (\boldsymbol{X}_j\boldsymbol{X}_k)^t \phi \right| \geq \lambda \right\}$$

$$\left| (\boldsymbol{X}_2\boldsymbol{X}_3)^T \phi \right| \geq \lambda$$

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0, 1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$$\mathcal{M}_\lambda = \left\{ \boxed{x_1 x_2} \quad \boxed{x_2 x_3} \right\}$$

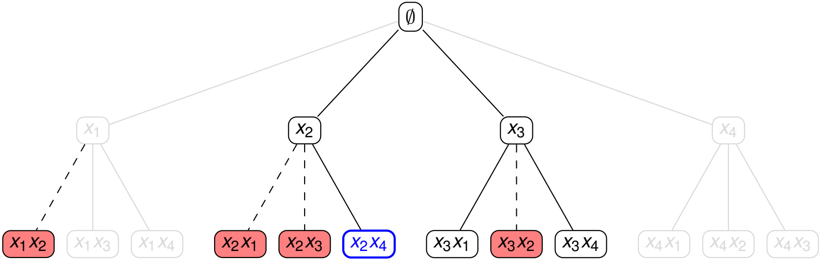
 # Identify violated constraints and update working set.
If $\eta(\phi, \boldsymbol{X}_j) \geq \lambda$, compute $\boldsymbol{m}_j(\phi)$ and update $\mathcal{M}_\lambda$.

$$\boldsymbol{m}_j(\phi) = \max_{\boldsymbol{X}_k :\, \boldsymbol{X}_j \boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| \left(\boldsymbol{X}_j \boldsymbol{X}_k \right)^T \phi \right|, \quad \mathcal{M}_\lambda \leftarrow \mathcal{M}_\lambda \cup \left\{ \boldsymbol{X}_j \boldsymbol{X}_k : \left| \left(\boldsymbol{X}_j \boldsymbol{X}_k \right)^t \phi \right| \geq \lambda \right\}$$

$$\left| \left(\boldsymbol{X}_2 \boldsymbol{X}_3 \right)^T \phi \right| \geq \lambda$$

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0,1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$$\mathcal{M}_\lambda = \left\{ \; x_1x_2 \quad x_2x_3 \; \right\}$$

 # Identify violated constraints and update working set.

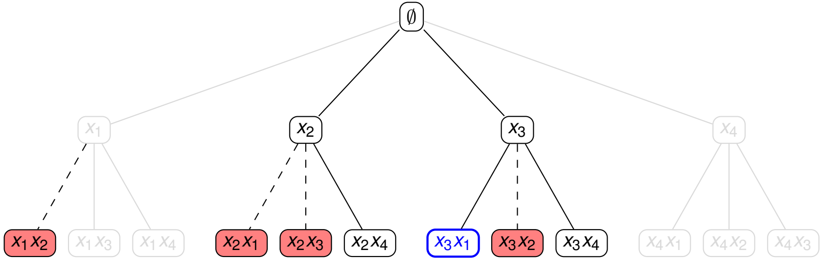
If $\eta(\phi, \boldsymbol{X}_j) \geq \lambda$, compute $\boldsymbol{m}_j(\phi)$ and update $\mathcal{M}_\lambda$.

$$\boldsymbol{m}_j(\phi) = \max_{\boldsymbol{X}_k : \, \boldsymbol{X}_j\boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| \left(\boldsymbol{X}_j\boldsymbol{X}_k\right)^T \phi \right|, \quad \mathcal{M}_\lambda \leftarrow \mathcal{M}_\lambda \cup \left\{ \boldsymbol{X}_j\boldsymbol{X}_k : \left| \left(\boldsymbol{X}_j\boldsymbol{X}_k\right)^t \phi \right| \geq \lambda \right\}$$

$$\left| \left(\boldsymbol{X}_2\boldsymbol{X}_4\right)^T \phi \right| < \lambda$$

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0, 1]\!]^{n\times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$$\mathcal{M}_\lambda = \left\{ \boxed{x_1 x_2} \quad \boxed{x_2 x_3} \right\}$$

 # Identify violated constraints and update working set.

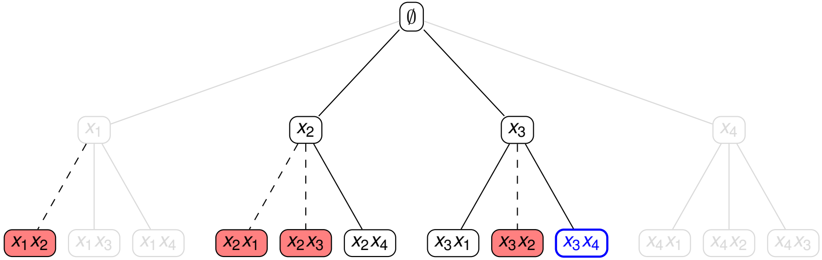
If $\eta(\phi, \boldsymbol{X}_j) \geq \lambda$, compute $\boldsymbol{m}_j(\phi)$ and update $\mathcal{M}_\lambda$.

$$\boldsymbol{m}_j(\phi) = \max_{\boldsymbol{X}_k:\, \boldsymbol{X}_j\boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| (\boldsymbol{X}_j\boldsymbol{X}_k)^T \phi \right|, \quad \mathcal{M}_\lambda \leftarrow \mathcal{M}_\lambda \cup \left\{ \boldsymbol{X}_j\boldsymbol{X}_k : \left| (\boldsymbol{X}_j\boldsymbol{X}_k)^t \phi \right| \geq \lambda \right\}$$

$$\left| (\boldsymbol{X}_3\boldsymbol{X}_1)^T \phi \right| < \lambda$$

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0, 1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



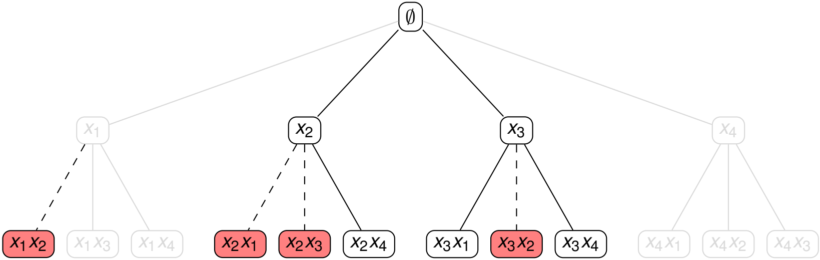
$$\mathcal{M}_\lambda = \left\{ \boxed{x_1 x_2} \quad \boxed{x_2 x_3} \right\}$$

 # Identify violated constraints and update working set.
If $\eta(\phi, \boldsymbol{X}_j) \geq \lambda$, compute $\boldsymbol{m}_j(\phi)$ and update $\mathcal{M}_\lambda$.

$$\boldsymbol{m}_j(\phi) = \max_{\boldsymbol{X}_k:\, \boldsymbol{X}_j\boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| (\boldsymbol{X}_j\boldsymbol{X}_k)^T \phi \right|, \quad \mathcal{M}_\lambda \leftarrow \mathcal{M}_\lambda \cup \left\{ \boldsymbol{X}_j\boldsymbol{X}_k : \left| (\boldsymbol{X}_j\boldsymbol{X}_k)^t \phi \right| \geq \lambda \right\}$$

$$\left| (\boldsymbol{X}_3\boldsymbol{X}_4)^T \phi \right| < \lambda$$

Input: $\boldsymbol{X} \in [\![0, 1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$$\mathcal{M}_\lambda = \left\{ \; x_1 x_2 \quad x_2 x_3 \; \right\}$$

 # Identify violated constraints and update working set.

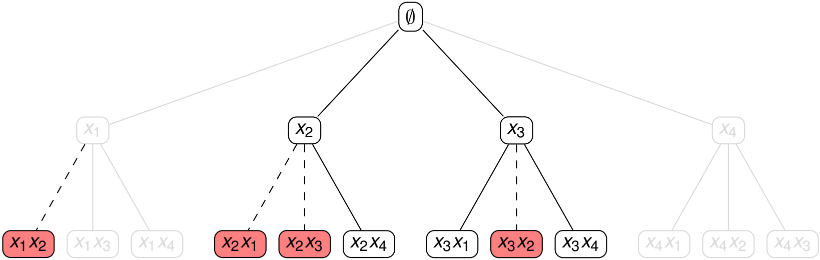
If $\eta(\phi, \boldsymbol{X}_j) \geq \lambda$, compute $\boldsymbol{m}_j(\phi)$ and update $\mathcal{M}_\lambda$.

$$\boldsymbol{m}_j(\phi) = \max_{\boldsymbol{X}_k : \, \boldsymbol{X}_j \boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| \left(\boldsymbol{X}_j \boldsymbol{X}_k \right)^T \phi \right|, \quad \mathcal{M}_\lambda \leftarrow \mathcal{M}_\lambda \cup \left\{ \boldsymbol{X}_j \boldsymbol{X}_k : \left| \left(\boldsymbol{X}_j \boldsymbol{X}_k \right)^t \phi \right| \geq \lambda \right\}$$

done

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0,1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$$\mathcal{M}_\lambda = \left\{ \boxed{x_1 x_2} \quad \boxed{x_2 x_3} \right\}$$

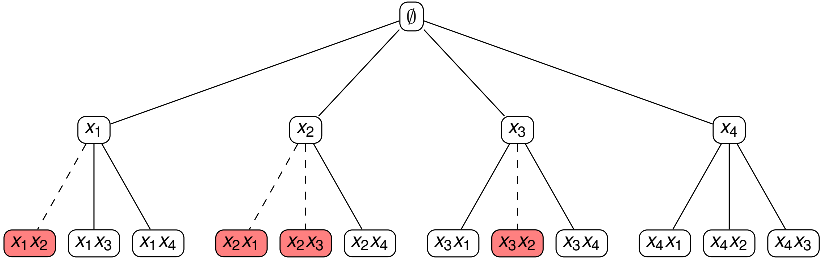
Solve subproblem restricted to $\mathcal{M}_\lambda$
Update residuals

$$\boldsymbol{w}, b \leftarrow \underset{(\boldsymbol{w},b) \in \mathbb{R}^D \times \mathbb{R}}{\operatorname{argmin}} \frac{1}{n} \|\boldsymbol{y} - \boldsymbol{Z}_{\mathcal{M}_\lambda} \boldsymbol{w} - b\|_2^2 + \lambda \|\boldsymbol{w}\|_1$$

$$\phi \leftarrow \boldsymbol{y} - \boldsymbol{Z}_{\mathcal{M}_\lambda} \boldsymbol{w} - b$$

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0,1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$

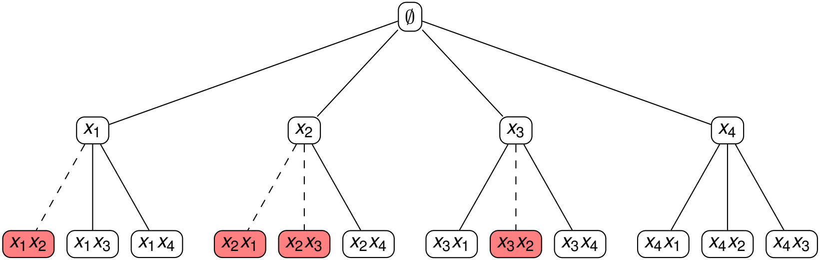


$$\mathcal{M}_\lambda = \left\{ \boxed{x_1 x_2} \quad \boxed{x_2 x_3} \right\}$$

Initialize $\mathcal{M}_\lambda$. Pre-solve and initialize ϕ.
repeat:
    for each branch j **do**:
        # Identify violated constraints and update working set.
        Compute the bound $\eta(\phi, \boldsymbol{X}_j)$.
        If $\eta(\phi, \boldsymbol{X}_j) \geq \lambda$, compute $\boldsymbol{m}_j(\phi)$ and update $\mathcal{M}_\lambda$.
    Solve subproblem
until no violated constraint remains.

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0,1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$

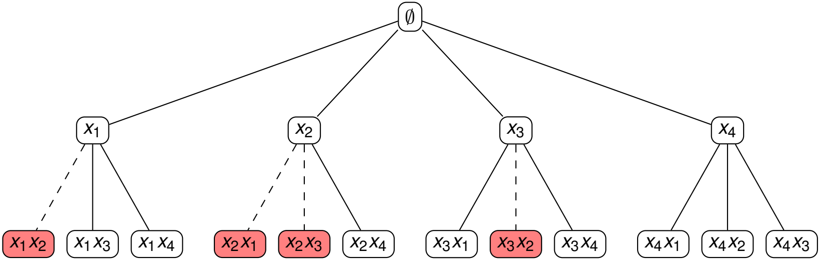


$$\mathcal{M}_\lambda = \left\{ \boxed{x_1 x_2} \quad \boxed{x_2 x_3} \right\}$$

Initialize $\mathcal{M}_\lambda$. Pre-solve and initialize ϕ.
repeat:
    for each branch j **do**:
        # Identify violated constraints and update working set.
        Compute the bound $\eta(\phi, \boldsymbol{X}_j)$.
        If $\eta(\phi, \boldsymbol{X}_j) \geq \lambda$, compute $\boldsymbol{m}_j(\phi)$ and update $\mathcal{M}_\lambda$.
    Solve subproblem
until no violated constraint remains.

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0,1]\!]^{n \times p}$, $\quad \boldsymbol{y} \in \mathbb{R}^n$



$$\mathcal{M}_\lambda = \left\{ \boxed{x_1 x_2} \quad \boxed{x_2 x_3} \right\}$$

Initialize $\mathcal{M}_\lambda$. Pre-solve and initialize ϕ.
repeat:
  for each branch j **do**:
    # Identify violated constraints and update working set.
    Compute the bound $\eta(\phi, \boldsymbol{X}_j)$.
    If $\eta(\phi, \boldsymbol{X}_j) \geq \lambda$, compute $\boldsymbol{m}_j(\phi)$ and update $\mathcal{M}_\lambda$.
  Solve subproblem
until no violated constraint remains.

Branch Bounds

- Suppose the current residual is ϕ. For each branch $j \in [\![1, p]\!]$, we want a bound $\eta(\phi, \boldsymbol{X}_j)$ such that:

$$\boldsymbol{m}_j(\phi) = \max_{\boldsymbol{X}_k :\, \boldsymbol{X}_j \boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| (\boldsymbol{X}_j \boldsymbol{X}_k)^T \phi \right| \leq \eta(\phi, \boldsymbol{X}_j)$$

- One possibility is to use the following bound (as in Nakagawa et al.):

$$\begin{aligned} \boldsymbol{m}_j(\phi) &\leq \max_{\boldsymbol{x} \in [\![0,1]\!]^n} \left| (\boldsymbol{X}_j \odot \boldsymbol{x})^T \phi \right| \\ &= \max \left(\sum_{i : \phi_i > 0} \boldsymbol{X}_{ij} \phi_i, - \sum_{i : \phi_i < 0} \boldsymbol{X}_{ij} \phi_i \right) \\ &= \zeta(\phi, \boldsymbol{X}_j) \end{aligned}$$

✓ can be computed very efficiently.
✓ but becomes too loose when n increases, leading to only few branches pruned.

Branch Bounds

- Suppose the current residual is ϕ.
  Suppose we have already computed $\boldsymbol{m}_j(\phi^{prev}) = \max_{\boldsymbol{X}_k:\, \boldsymbol{X}_j\boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left|(\boldsymbol{X}_j\boldsymbol{X}_k)^T \phi^{prev}\right|$

  We propose the following bound:

$$
\begin{aligned}
\boldsymbol{m}_j(\phi) &= \max_{\boldsymbol{x}\in\mathcal{D}e(\boldsymbol{X}_j)} \left|\boldsymbol{x}^T\phi\right| \\
&= \max_{\boldsymbol{x}\in\mathcal{D}e(\boldsymbol{X}_j)} \left|\boldsymbol{x}^T\phi^{prev} + \boldsymbol{x}^T(\phi - \phi^{prev})\right| \\
&\leq \max_{\boldsymbol{x}\in\mathcal{D}e(\boldsymbol{X}_j)} \left|\boldsymbol{x}^T\phi^{prev}\right| + \max_{\boldsymbol{x}\in\mathcal{D}e(\boldsymbol{X}_j)} \left|\boldsymbol{x}^T(\phi - \phi^{prev})\right| \\
&\leq \boldsymbol{m}_j(\phi_{prev}) + \max_{\boldsymbol{x}\in[\![0,1]\!]^n} \left|(\boldsymbol{X}_j \odot \boldsymbol{x})^T(\phi - \phi^{prev})\right| \\
&= \boldsymbol{m}_j(\phi_{prev}) + max\left(\sum_{i:\phi_i > \phi_i^{prev}} \boldsymbol{X}_{ij}(\phi_i - \phi_i^{prev}), - \sum_{i:\phi_i < \phi_i^{prev}} \boldsymbol{X}_{ij}(\phi_i - \phi_i^{prev})\right) \\
&= \eta(\phi, \boldsymbol{X}_j)
\end{aligned}
$$

- We leverage previously computed maximum inner products and the fact that residuals along the regularization path are correlated.

Branch Bounds

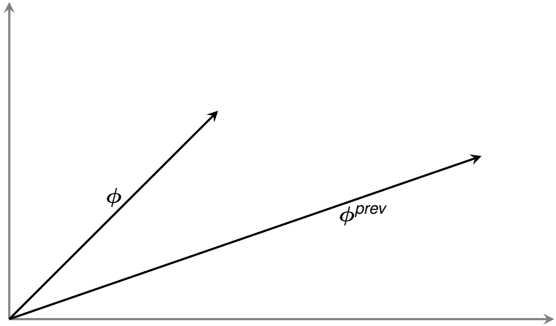
- Suppose the current residual is ϕ.
  Suppose we have already computed $\boldsymbol{m}_j(\phi^{prev}) = \max_{\boldsymbol{X}_k:\, \boldsymbol{X}_j\boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left|(\boldsymbol{X}_j\boldsymbol{X}_k)^T \phi^{prev}\right|$
  We propose the following bound:

$$
\begin{aligned}
\boldsymbol{m}_j(\phi) &= \max_{\boldsymbol{x}\in\mathcal{D}e(\boldsymbol{X}_j)} \left|\boldsymbol{x}^T\phi\right| \\
&= \max_{\boldsymbol{x}\in\mathcal{D}e(\boldsymbol{X}_j)} \left|\alpha\boldsymbol{x}^T\phi^{prev} + \boldsymbol{x}^T\left(\phi - \alpha\phi^{prev}\right)\right| \\
&\leq \max_{\boldsymbol{x}\in\mathcal{D}e(\boldsymbol{X}_j)} |\alpha| \left|\boldsymbol{x}^T\phi^{prev}\right| + \max_{\boldsymbol{x}\in\mathcal{D}e(\boldsymbol{X}_j)} \left|\boldsymbol{x}^T\left(\phi - \alpha\phi^{prev}\right)\right| \\
&\leq |\alpha|\, \boldsymbol{m}_j(\phi_{prev}) + \max_{\boldsymbol{x}\in[\![0,1]\!]^n} \left|\left(\boldsymbol{X}_j \odot \boldsymbol{x}\right)^T\left(\phi - \alpha\phi^{prev}\right)\right| \\
&= |\alpha|\, \boldsymbol{m}_j(\phi_{prev}) + max\left(\sum_{i:\phi_i > \alpha\phi_i^{prev}} \boldsymbol{X}_{ij}\left(\phi_i - \alpha\phi_i^{prev}\right), -\sum_{i:\phi_i < \alpha\phi_i^{prev}} \boldsymbol{X}_{ij}\left(\phi_i - \alpha\phi_i^{prev}\right)\right) \\
&= \eta(\phi, \boldsymbol{X}_j, \alpha)
\end{aligned}
$$

- We leverage previously computed maximum inner products and the fact that residuals along the regularization path are correlated.

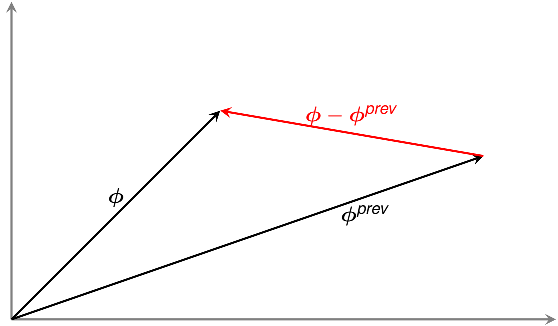
Branch Bounds: Intuition

$$\eta(\phi, \boldsymbol{X}_j, \alpha) = |\alpha|\, \boldsymbol{m}_j + max\left(\sum_{i:\phi_i > \alpha\phi_i^{prev}} \boldsymbol{X}_{ij}\left(\phi_i - \alpha\phi_i^{prev}\right), - \sum_{i:\phi_i < \alpha\phi_i^{prev}} \boldsymbol{X}_{ij}\left(\phi_i - \alpha\phi_i^{prev}\right) \right)$$



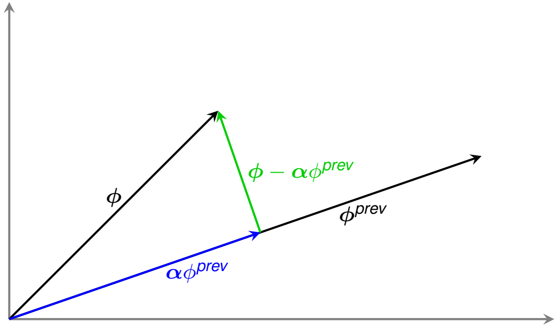
Branch Bounds: Intuition

$$\eta(\phi, \boldsymbol{X}_j, \alpha) = |\alpha|\, \boldsymbol{m}_j + max \left(\sum_{i:\phi_i > \alpha\phi_i^{prev}} \boldsymbol{X}_{ij} \left(\phi_i - \alpha\phi_i^{prev}\right), - \sum_{i:\phi_i < \alpha\phi_i^{prev}} \boldsymbol{X}_{ij} \left(\phi_i - \alpha\phi_i^{prev}\right) \right)$$



Branch Bounds: Intuition

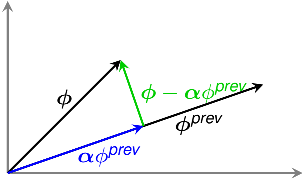
$$\eta(\phi, \boldsymbol{X}_j, \alpha) = |\alpha| \, \boldsymbol{m}_j + max \left(\sum_{i:\phi_i > \alpha \phi_i^{prev}} \boldsymbol{X}_{ij} \left(\phi_i - \alpha \phi_i^{prev}\right), - \sum_{i:\phi_i < \alpha \phi_i^{prev}} \boldsymbol{X}_{ij} \left(\phi_i - \alpha \phi_i^{prev}\right) \right)$$



Branch Bounds: Optimisation

$$\eta(\phi, \boldsymbol{X}_j, \alpha) = |\alpha|\, \boldsymbol{m}_j + max\left(\sum_{i:\phi_i > \alpha\phi_i^{prev}} \boldsymbol{X}_{ij}\left(\phi_i - \alpha\phi_i^{prev}\right), -\sum_{i:\phi_i < \alpha\phi_i^{prev}} \boldsymbol{X}_{ij}\left(\phi_i - \alpha\phi_i^{prev}\right)\right)$$

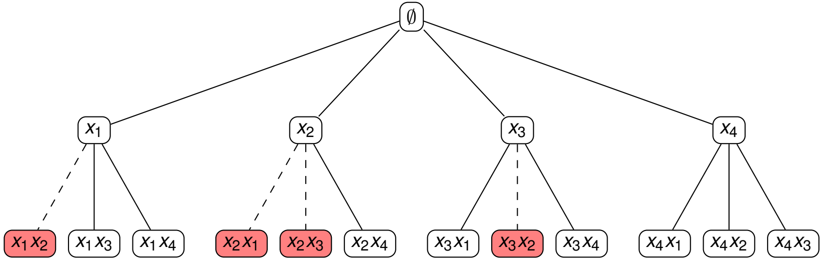
How to choose α?



- Option 1: $\alpha^* = \underset{\alpha\in\mathbb{R}}{\operatorname{argmin}}\, \eta(\phi, \boldsymbol{X}_j, \alpha)$
  - ✓ η is a piecewise continuous function which is convex in α.
  - ✓ η can be minimized in $\mathcal{O}(n_j \log n_j)$ operations.
- Option 2: $\alpha_{\ell 2} = \frac{\phi^T \phi^{prev}}{\|\phi^{prev}\|_2^2}$
  - ✓ $\alpha_{\ell 2}$ minimizes $\|\phi - \alpha\phi^{prev}\|_2^2$.
  - ✓ $\alpha_{\ell 2}$ can be obtained in $\mathcal{O}(n_j)$ operations.

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0,1]\!]^{n\times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$

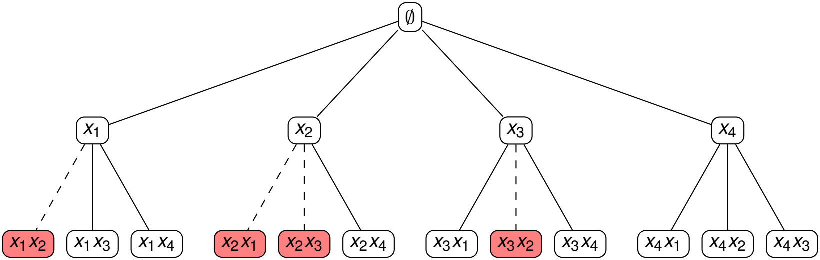


$$\mathcal{M}_\lambda = \left\{ \boxed{x_1 x_2} \quad \boxed{x_2 x_3} \right\}$$

Initialize $\mathcal{M}_\lambda$
repeat:
    for each branch j **do**:
        # Identify violated constraints and update working set.
        Compute the bound $\eta(\phi, \boldsymbol{X}_j)$.
        If $\eta(\phi, \boldsymbol{X}_j) \geq \lambda$, compute $\boldsymbol{m}_j(\phi)$ and update $\mathcal{M}_\lambda$.
    Solve subproblem
until no violated constraint remains.

WHInter pseudo algorithm

Input: $\boldsymbol{X} \in [\![0, 1]\!]^{n \times p}, \quad \boldsymbol{y} \in \mathbb{R}^n$



$$\mathcal{M}_\lambda = \left\{ \boxed{x_1 x_2} \quad \boxed{x_2 x_3} \right\}$$

Initialize $\mathcal{M}_\lambda$
repeat:
    for each branch j **do**:
        # Identify violated constraints and update working set.
        Compute the bound $\eta(\phi, \boldsymbol{X}_j)$.
        If $\eta(\phi, \boldsymbol{X}_j) \geq \lambda$, compute $\boldsymbol{m}_j(\phi)$ and update $\mathcal{M}_\lambda$.
    Solve subproblem
until no violated constraint remains.

Maximum inner product computation

- Whenever $\eta(\phi, \boldsymbol{X}_j, \alpha^*) \geq \lambda$, then there is a chance that branch j contains a feature which is violated for the current residual ϕ. In this case we need to:
  - ✓ identify all violated features.
  - ✓ compute:

$$\boldsymbol{m}_j = \max_{\boldsymbol{X}_k : \boldsymbol{X}_j \boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| (\boldsymbol{X}_j \odot \boldsymbol{X}_k)^T \phi \right|$$

  which naively requires computing all inner products.

- We note that $\boldsymbol{m}_j$ can be rewritten as:

$$\boldsymbol{m}_j = \max_{\boldsymbol{X}_k : \boldsymbol{X}_j \boldsymbol{X}_k \notin \mathcal{M}_\lambda} \left| \boldsymbol{X}_k^T (\boldsymbol{X}_j \odot \phi) \right|$$

  This is (almost) a Maximum Inner product Search (MIPS) problem with query vector $\boldsymbol{X}_j \odot \phi$ and database or probe vectors $\{\boldsymbol{X}_k, k \in [\![1, p]\!] : \boldsymbol{X}_j \boldsymbol{X}_k \notin \mathcal{M}_\lambda\}$.

Maximum Inner Product Search

- Relevant work in the data mining literature (far from exhaustive):
  - ✓ State-of-the-art exact MIPS algorithm:

    Christina Teflioudi and Rainer Gemulla. "Exact and Approximate Maximum Inner Product Search with LEMP". . In: *TODS* (2016)
  - ✓ All pairs similarity search algorithm:

    Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. "Scaling up all pairs similarity search". In: *Proc. 16th Int. Conf. World Wide Web - WWW '07* (2007), p. 131

- We borrow the idea of computing inner products on restricted sets of dimensions, and bounding the part of the inner product on the remaining dimensions.

- We implement this idea in the case of our particular setting where:
  - ✓ queries are sparse vectors of the form: $\boldsymbol{X}_j \odot \phi$, and ϕ can have both positive and negative entries.
  - ✓ probes are binary vectors.

Input: $\boldsymbol{Q} = \{\boldsymbol{X}_j : \eta(\phi, \boldsymbol{X}_j) \geq \lambda\} \in [0,1]^{n \times q}$, $\boldsymbol{P} \in [0,1]^{n \times p}$, $\boldsymbol{\phi} \in \mathbb{R}^n$
Param: $n_c \in \mathbb{N}$
Output: $\boldsymbol{m} \in \mathbb{R}^q$ and $\boldsymbol{k} \in \mathbb{R}^q$.

```
Reorder the dimensions 1 ... n such that |φ| is sorted in descending order.
Reorder the vectors P_j in P in increasing order of nnz(P_j).
Initialize the best inner products m ∈ R^q for each query.
We note N_j ⊂ [[1, n]] the set of non zero entries of X_j.
for j ∈ [[1, q]] do
    # Compute r+ ∈ R^n and r- ∈ R^n the partial inner product upperbounds.
    for i ∈ N_j do
        r+_i = Σ_{m>i; m:φ_m>0} X_mj φ_m   and   r-_i = Σ_{m>i; m:φ_m<0} X_mj φ_m
    for k ∈ [[1, p]] do
        d = 0 (inner product initialization); c = 0 (counter initialization);
        for i ∈ N_j do
            d = d + Q_ij P_ik φ_i;   c = c + 1.
            if c mod n_c == 0 then
                if (d + r+_i) < min(m_j, λ) and |(d + r-_i)| < min(m_j, λ) then go to next probe.
        if m_j < d < λ then set m_j = d and k_j = k
        if d ≥ λ then add X_j X_k to M_λ
```

Maximum Inner Product Search

Input: $\boldsymbol{Q} = \{\boldsymbol{X}_j : \eta(\phi, \boldsymbol{X}_j) \geq \lambda\} \in [0,1]^{n \times q}$, $\boldsymbol{P} \in [0,1]^{n \times p}$, $\phi \in \mathbb{R}^n$
Param: $n_c \in \mathbb{N}$
Output: $\boldsymbol{m} \in \mathbb{R}^q$ and $\boldsymbol{k} \in \mathbb{R}^q$.

Reorder the dimensions $1 \ldots n$ such that $|\phi|$ is sorted in descending order.
Reorder the vectors $\boldsymbol{P}_j$ in $\boldsymbol{P}$ in increasing order of $\mathtt{nnz}(\boldsymbol{P}_j)$.
Initialize the best inner products $\boldsymbol{m} \in \mathbb{R}^q$ for each query.
We note $\mathcal{N}_j \subset [\![1, n]\!]$ the set of non zero entries of $\boldsymbol{X}_j$.
for $j \in [\![1, q]\!]$ **do**
    # Compute $\boldsymbol{r}^+ \in \mathbb{R}^n$ and $\boldsymbol{r}^- \in \mathbb{R}^n$ the partial inner product upperbounds.
    for $i \in \mathcal{N}_j$ **do**

$$\boldsymbol{r}_i^+ = \sum_{m>i;\ m:\phi_m>0} \boldsymbol{X}_{mj}\phi_m \quad \text{and} \quad \boldsymbol{r}_i^- = \sum_{m>i;\ m:\phi_m<0} \boldsymbol{X}_{mj}\phi_m$$

    for $k \in [\![1, p]\!]$ **do**
        d = 0 (inner product initialization); c = 0 (counter initialization);
        for $i \in \mathcal{N}_j$ **do**
            $d = d + \boldsymbol{Q}_{ij}\boldsymbol{P}_{ik}\phi_i; \quad c = c + 1.$
            if $c \mod n_c == 0$ **then**
                if $(d + \boldsymbol{r}_i^+) < \min(\boldsymbol{m}_j, \lambda)$ and $\left|(d + \boldsymbol{r}_i^-)\right| < \min(\boldsymbol{m}_j, \lambda)$ **then** go to next probe.
        if $\boldsymbol{m}_j < d < \lambda$ **then** set $\boldsymbol{m}_j = d$ and $\boldsymbol{k}_j = k$
        if $d \geq \lambda$ **then** add $\boldsymbol{X}_j\boldsymbol{X}_k$ to $\mathcal{M}_\lambda$

Input: $\boldsymbol{Q} = \{\boldsymbol{X}_j : \eta(\phi, \boldsymbol{X}_j) \geq \lambda\} \in [0,1]^{n \times q}$, $\quad \boldsymbol{P} \in [0,1]^{n \times p}$, $\quad \phi \in \mathbb{R}^n$
Param: $n_c \in \mathbb{N}$
Output: $\boldsymbol{m} \in \mathbb{R}^q$ and $\boldsymbol{k} \in \mathbb{R}^q$.

Reorder the dimensions $1 \ldots n$ such that $|\phi|$ is sorted in descending order.
Reorder the vectors $\boldsymbol{P}_j$ in $\boldsymbol{P}$ in increasing order of $\mathtt{nnz}(\boldsymbol{P}_j)$.
Initialize the best inner products $\boldsymbol{m} \in \mathbb{R}^q$ for each query.
We note $\mathcal{N}_j \subset [\![1, n]\!]$ the set of non zero entries of $\boldsymbol{X}_j$.
for $j \in [\![1, q]\!]$ **do**
  # Compute $\boldsymbol{r}^+ \in \mathbb{R}^n$ and $\boldsymbol{r}^- \in \mathbb{R}^n$ the partial inner product upperbounds.
  for $i \in \mathcal{N}_j$ **do**

$$\boldsymbol{r}_i^+ = \sum_{m>i;\ m:\phi_m>0} \boldsymbol{X}_{mj}\phi_m \quad \text{and} \quad \boldsymbol{r}_i^- = \sum_{m>i;\ m:\phi_m<0} \boldsymbol{X}_{mj}\phi_m$$

  for $k \in [\![1, p]\!]$ **do**
    d = 0 (inner product initialization); c = 0 (counter initialization);
    for $i \in \mathcal{N}_j$ **do**
      $d = d + \boldsymbol{Q}_{ij}\boldsymbol{P}_{ik}\phi_i; \quad c = c + 1.$
      if $c \mod n_c == 0$ **then**
        if $(d + \boldsymbol{r}_i^+) < \min(\boldsymbol{m}_j, \lambda)$ and $\left|(d + \boldsymbol{r}_i^-)\right| < \min(\boldsymbol{m}_j, \lambda)$ **then** go to next probe.
    if $\boldsymbol{m}_j < d < \lambda$ **then** set $\boldsymbol{m}_j = d$ and $\boldsymbol{k}_j = k$
    if $d \geq \lambda$ **then** add $\boldsymbol{X}_j\boldsymbol{X}_k$ to $\mathcal{M}_\lambda$

Maximum Inner Product Search

Input: $\boldsymbol{Q} = \{\boldsymbol{X}_j : \eta(\phi, \boldsymbol{X}_j) \geq \lambda\} \in [0,1]^{n \times q}$, $\boldsymbol{P} \in [0,1]^{n \times p}$, $\phi \in \mathbb{R}^n$
Param: $n_c \in \mathbb{N}$
Output: $\boldsymbol{m} \in \mathbb{R}^q$ and $\boldsymbol{k} \in \mathbb{R}^q$.

Reorder the dimensions $1 \ldots n$ such that $|\phi|$ is sorted in descending order.
Reorder the vectors $\boldsymbol{P}_j$ in $\boldsymbol{P}$ in increasing order of $\mathtt{nnz}(\boldsymbol{P}_j)$.
Initialize the best inner products $\boldsymbol{m} \in \mathbb{R}^q$ for each query.
We note $\mathcal{N}_j \subset [\![1, n]\!]$ the set of non zero entries of $\boldsymbol{X}_j$.
for $j \in [\![1, q]\!]$ **do**
 # Compute $\boldsymbol{r}^+ \in \mathbb{R}^n$ and $\boldsymbol{r}^- \in \mathbb{R}^n$ the partial inner product upperbounds.
for $i \in \mathcal{N}_j$ **do**

$$\boldsymbol{r}_i^+ = \sum_{m>i;\ m:\phi_m>0} \boldsymbol{X}_{mj}\phi_m \quad \text{and} \quad \boldsymbol{r}_i^- = \sum_{m>i;\ m:\phi_m<0} \boldsymbol{X}_{mj}\phi_m$$

for $k \in [\![1, p]\!]$ **do**
d = 0 (inner product initialization); c = 0 (counter initialization);
for $i \in \mathcal{N}_j$ **do**
$d = d + \boldsymbol{Q}_{ij}\boldsymbol{P}_{ik}\phi_i$; $c = c + 1$.
if $c \mod n_c == 0$ **then**
if $(d + \boldsymbol{r}_i^+) < \min(\boldsymbol{m}_j, \lambda)$ $\mathtt{and}$ $\left|(d + \boldsymbol{r}_i^-)\right| < \min(\boldsymbol{m}_j, \lambda)$ **then** go to next probe.
if $\boldsymbol{m}_j < d < \lambda$ **then** set $\boldsymbol{m}_j = d$ and $\boldsymbol{k}_j = k$
if $d \geq \lambda$ **then** add $\boldsymbol{X}_j\boldsymbol{X}_k$ to $\mathcal{M}_\lambda$

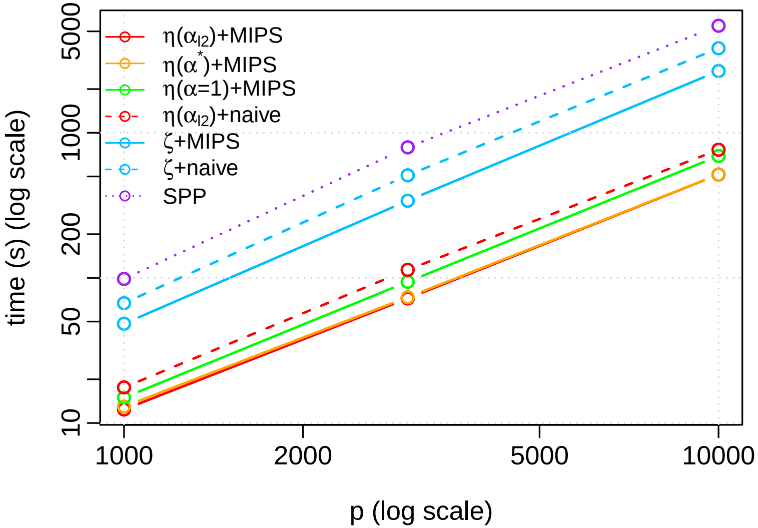
- We propose **WHInter**:
  - ✓ Working set strategy.
  - ✓ New branch pruning strategy for the identification of the active set.
  - ✓ Efficient computation of branch bounds using a Maximum Inner Product Search (MIPS) framework for binary data.
- Evaluation of WHInter on:
  - ✓ Simulated datasets.
  - ✓ Real toxicogenetics dataset.

LASSO Simulations

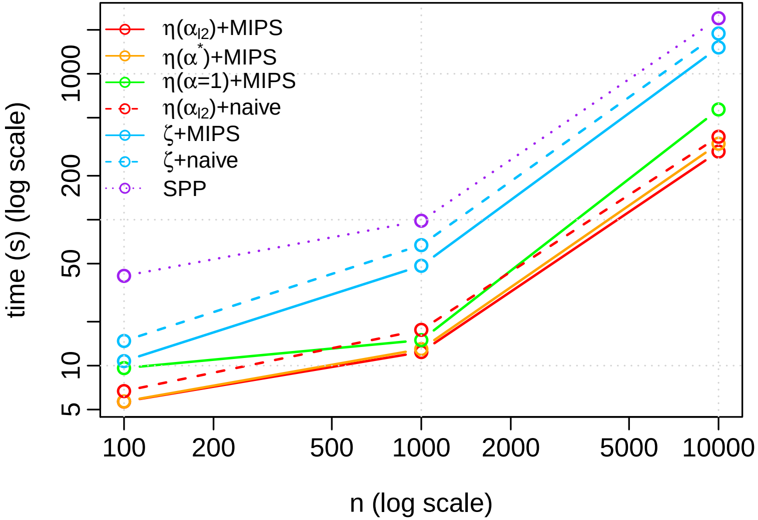
- We simulate $\boldsymbol{X} \in [\![0, 1]\!]^{n \times p}$ where $\boldsymbol{X}_{ik} \sim \text{Bernoulli}\,(q_k)$, and $q_k \sim \text{Unif}\,(0.1, 0.5)$ for different combinations of n and p.
- We randomly pick a set $\mathcal{S}$ of 100 features (among original and interaction ones).
- The associated weights are drawn from a standard gaussian distribution.
- We set $\phi = \boldsymbol{Z}_{\mathcal{S}}\boldsymbol{w}$
- We choose 100 values of λ logarithmically spaced in $[\lambda_{max}, 0.01\lambda_{max}]$. We stop the algirithm as soon as more than 150 features are selected in the model.

LASSO Simulations

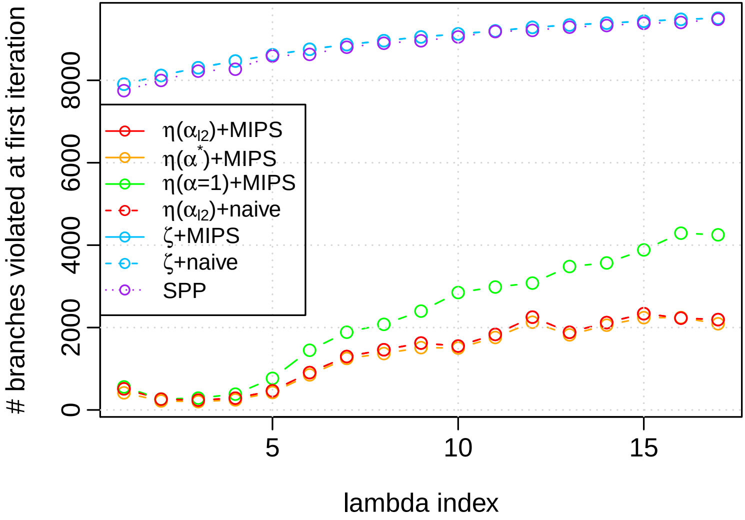
$n = 1000$ fixed, p varied.



$p = 1000$ fixed, n varied.



$n = 1000, p = 10000.$



Results on Dream 8 toxicogenetics data

- We consider the SNPs from chromosomes 1 and 19 of 620 lymphoblastoid cell lines, represented by $\boldsymbol{X}^1 \in [\![0,1]\!]^{620\times 102196}$ and $\boldsymbol{X}^{19} \in [\![0,1]\!]^{620\times 28418}$.
- The response $\boldsymbol{y} \in \mathbb{R}^{620}$ is the cytotoxicity (EC10) of a chemical compound.
- Correction for population structure was applied as in Price et al (2006).

Method	Chromosome 19 Preproc (min)	Chromosome 19 Path (min)	Chromosome 19 Tot. time (min)	Chromosome 1 Preproc (h)	Chromosome 1 Path (h)	Chromosome 1 Tot. time (h)
$\eta(\alpha_{l2}) + \mathit{MIPS}$	13	13	26	2.5	1.4	3.9
$\eta(\alpha^*) + \mathit{MIPS}$	13	13	26	2.5	1.2	3.7
$\eta(\alpha = 1) + \mathit{MIPS}$	13	22	35	2.5	2.9	5.4
$\eta(\alpha_{l2}) + \mathit{naive}$	13	23	36	2.5	2.5	5
$\zeta + \mathit{MIPS}$	7	84	91	1.2	13.5	14.7
$\zeta + \mathit{naive}$	7	109	116	1.2	17.2	18.4
SPP	7	173	180	1.2	25.2	26.4

Thank you for your attention.

- Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. "Scaling up all pairs similarity search". In: *Proc. 16th Int. Conf. World Wide Web - WWW '07* (2007), p. 131.
- Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. "Mind the duality gap: safer rules for the Lasso". In: *ICML*. 2015, pp. 333–342.
- Kazuya Nakagawa et al. "Safe Pattern Pruning: An Efficient Approach for Predictive Pattern Mining". In: *KDD* (2016). arXiv: 1602.04548.
- Christina Teflioudi and Rainer Gemulla. "Exact and Approximate Maximum Inner Product Search with LEMP". In: *TODS* (2016).