The Trimmed Lasso:

Sparse recovery guarantees and practical optimization by the Generalized Soft-Min Penalty

Boaz Nadler

Weizmann Institute of Science

Joint work Tal Amir and Ronen Basri

Statistics in the Big Data Era

June 2022

Peter's Non-Sparse Influence on My Work

- Some theory for Fisher's LDA... when there are many more variables than observations, 2004'

Peter's Non-Sparse Influence on My Work

- Some theory for Fisher's LDA... when there are many more variables than observations, 2004'

$\rightarrow$ The prediction error in PLS and CLS, 05'

Peter's Non-Sparse Influence on My Work

- Some theory for Fisher's LDA... when there are many more variables than observations, 2004'

$\rightarrow$ The prediction error in PLS and CLS, 05'

- Covariance Regularization by Thresholding, 08'

$\rightarrow$ Minimax bounds on sparse PCA,

Peter's Non-Sparse Influence on My Work

- Some theory for Fisher's LDA... when there are many more variables than observations, 2004'

$\rightarrow$ The prediction error in PLS and CLS, 05'

- Covariance Regularization by Thresholding, 08'

$\rightarrow$ Minimax bounds on sparse PCA,

- Simultaneous analysis of Lasso and Dantzig, 09'

Peter's Non-Sparse Influence on My Work

- Some theory for Fisher's LDA... when there are many more variables than observations, 2004'

$\rightarrow$ The prediction error in PLS and CLS, 05'

- Covariance Regularization by Thresholding, 08'

$\rightarrow$ Minimax bounds on sparse PCA,

- Simultaneous analysis of Lasso and Dantzig, 09'

Today's talk: Sparse Linear Regression

Problem setup:

Observe

(i) $n \times d$ matrix A

(ii) response vector $\boldsymbol{y} \in \mathbb{R}^n$

Problem setup:

Observe

(i) $n \times d$ matrix A

(ii) response vector $\boldsymbol{y} \in \mathbb{R}^n$

Given sparsity parameter k

Problem setup:

Observe

(i) $n \times d$ matrix A

(ii) response vector $\boldsymbol{y} \in \mathbb{R}^n$

Given sparsity parameter k

solve

$$\boxed{\min_{\boldsymbol{x}} \|A\boldsymbol{x} - \boldsymbol{y}\|_2 \quad \text{subject to } \|\boldsymbol{x}\|_0 \leq k} \tag{P0}$$

Sparse Approximation

$$\boxed{\min_{\boldsymbol{x}} \|A\boldsymbol{x} - \boldsymbol{y}\|_2 \quad \text{subject to } \|\boldsymbol{x}\|_0 \leq k} \tag{P0}$$

Sparse Approximation

$$\boxed{\min_{\boldsymbol{x}} \|A\boldsymbol{x} - \boldsymbol{y}\|_2 \quad \text{subject to } \|\boldsymbol{x}\|_0 \leq k} \qquad \text{(P0)}$$

Signal/Image processing:
$\boldsymbol{y} = (y_1, \ldots, y_n)$ are n samples of unknown function
A = dictionary, whose columns are basic signals / atoms

Sparse Approximation

$$\boxed{\min_{\boldsymbol{x}} \|A\boldsymbol{x} - \boldsymbol{y}\|_2 \quad \text{subject to } \|\boldsymbol{x}\|_0 \le k} \qquad \text{(P0)}$$

Signal/Image processing:

$\boldsymbol{y} = (y_1, \dots, y_n)$ are n samples of unknown function

A = dictionary, whose columns are basic signals / atoms

Seek best representation of $\boldsymbol{y}$ by at most k *dictionary atoms*.

Sparse Approximation

$$\boxed{\min_{\boldsymbol{x}} \|A\boldsymbol{x} - \boldsymbol{y}\|_2 \quad \text{subject to } \|\boldsymbol{x}\|_0 \leq k} \qquad \text{(P0)}$$

Signal/Image processing:

$\boldsymbol{y} = (y_1, \ldots, y_n)$ are n samples of unknown function

$A =$ dictionary, whose columns are basic signals / atoms

Seek best representation of $\boldsymbol{y}$ by at most k *dictionary atoms*.

Compressed sensing:

Wish to recover unknown signal $\boldsymbol{x} \in \mathbb{R}^d$, from n noisy observations

$$y_i = \mathbf{w}_i^\top \boldsymbol{x} + \sigma \xi_i$$

Assume that $\boldsymbol{x}$ is (approximately) k-sparse

Sparse Approximation

Statistics: sparse linear regression

given n observations (X_i, y_i), assumed of the form

$$y = X^\top \beta + \varepsilon$$

y is a response variable that we wish to predict from an explanatory vector $X \in \mathbb{R}^d$

Sparse Approximation

Statistics: sparse linear regression

given n observations (X_i, y_i), assumed of the form

$$y = X^\top \beta + \varepsilon$$

y is a response variable that we wish to predict from an explanatory vector $X \in \mathbb{R}^d$

...using at most k explanatory variables.

Sparsity parameter k

Often k is unknown and needs to be estimated

Sparsity parameter k

Often k is unknown and needs to be estimated

A common approach: Solve (P0) for several values of k and apply:

- Cross validation
- Model selection criterion

Sparsity parameter k

Often k is unknown and needs to be estimated

A common approach: Solve (P0) for several values of k and apply:

- Cross validation
- Model selection criterion

In rest of talk: Assume k is given

Sparsity parameter k

Often k is unknown and needs to be estimated

A common approach: Solve (P0) for several values of k and apply:

- Cross validation
- Model selection criterion

In rest of talk: Assume k is given

Focus on solving (P0) for a *given* value of k

Support Detection

The key challenge in solving (P0) is *support detection*, finding the optimal k columns of A to include in the solution

Support Detection

The key challenge in solving (P0) is *support detection*, finding the optimal k columns of A to include in the solution

Once support has been found, problem reduces to solving least squares on these k columns.

Support Detection

The key challenge in solving (P0) is *support detection*, finding the optimal k columns of A to include in the solution

Once support has been found, problem reduces to solving least squares on these k columns.

[Natarajan 95', Davis et al 97']

Unfortunately, this problem is NP-hard...

Support Detection

The key challenge in solving (P0) is *support detection*, finding the optimal k columns of A to include in the solution

Once support has been found, problem reduces to solving least squares on these k columns.

[Natarajan 95', Davis et al 97']

Unfortunately, this problem is NP-hard...

Yet, extensive *prior work*, on algorithms, theory, lower bounds, etc.

Support Detection

The key challenge in solving (P0) is *support detection*, finding the optimal k columns of A to include in the solution

Once support has been found, problem reduces to solving least squares on these k columns.

[Natarajan 95', Davis et al 97']

Unfortunately, this problem is NP-hard...

Yet, extensive *prior work*, on algorithms, theory, lower bounds, etc.

Over a hundred methods to approximately solve (P0)

Support Detection

The key challenge in solving (P0) is *support detection*, finding the optimal k columns of A to include in the solution

Once support has been found, problem reduces to solving least squares on these k columns.

[Natarajan 95', Davis et al 97']

Unfortunately, this problem is NP-hard...

Yet, extensive *prior work*, on algorithms, theory, lower bounds, etc.

Over a hundred methods to approximately solve (P0)
lots of theoretical results, recovery guarantees, etc.

(Almost) all prior work on (P0) in 3 slides...

Previous Work

Greedy methods:

- Matching Pursuit algorithms
  - Orthogonal Matching Pursuit (OMP), CoSaMP [Needell, Tropp, ACHA 2009] and more
- Iterative Hard Thresholding [Blumensath, Davies, ACHA 2009]
- Iterative Support Detection (ISD) [Wang, Yin, Im. Sc. 2010]
- Forward stepwise linear regression (1960's), etc.

Previous Work

Greedy methods:

- Matching Pursuit algorithms
  - Orthogonal Matching Pursuit (OMP), CoSaMP [Needell, Tropp, ACHA 2009] and more
- Iterative Hard Thresholding [Blumensath, Davies, ACHA 2009]
- Iterative Support Detection (ISD) [Wang, Yin, Im. Sc. 2010]
- Forward stepwise linear regression (1960's), etc.

Advantages: Easy to program, run very fast.

Previous Work

Greedy methods:

- Matching Pursuit algorithms
  - Orthogonal Matching Pursuit (OMP), CoSaMP [Needell, Tropp, ACHA 2009] and more
- Iterative Hard Thresholding [Blumensath, Davies, ACHA 2009]
- Iterative Support Detection (ISD) [Wang, Yin, Im. Sc. 2010]
- Forward stepwise linear regression (1960's), etc.

Advantages: Easy to program, run very fast.

Limitation: May yield suboptimal solutions.

Penalty Methods

Replace constraint $\|\boldsymbol{x}\|_0 \leq k$ by a penalty $\rho(\boldsymbol{x})$:

$$\min_{\boldsymbol{x}} \ \tfrac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\rho(\boldsymbol{x}).$$

Penalty Methods

Replace constraint $\|\boldsymbol{x}\|_0 \leq k$ by a penalty $\rho(\boldsymbol{x})$:

$$\min_{\boldsymbol{x}} \; \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\rho(\boldsymbol{x}).$$

- To obtain a k-sparse solution, λ needs to be tuned.

Penalty Methods

Replace constraint $\|\boldsymbol{x}\|_0 \leq k$ by a penalty $\rho(\boldsymbol{x})$:

$$\min_{\boldsymbol{x}} \; \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\rho(\boldsymbol{x}).$$

- To obtain a k-sparse solution, λ needs to be tuned.

The most popular penalty is the convex *lasso*: $\rho(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$

Penalty Methods

Replace constraint $\|\boldsymbol{x}\|_0 \leq k$ by a penalty $\rho(\boldsymbol{x})$:

$$\min_{\boldsymbol{x}} \ \tfrac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\rho(\boldsymbol{x}).$$

- To obtain a k-sparse solution, λ needs to be tuned.

The most popular penalty is the convex *lasso*: $\rho(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$

Lasso:

- Recovery guarantees under various conditions (Incoherence, RIP, Restricted Eigenvalue, ...)
- Fast optimization schemes developed

Penalty Methods

Replace constraint $\|\boldsymbol{x}\|_0 \le k$ by a penalty $\rho(\boldsymbol{x})$:

$$\min_{\boldsymbol{x}} \ \tfrac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\rho(\boldsymbol{x}).$$

- To obtain a k-sparse solution, λ needs to be tuned.

The most popular penalty is the convex *lasso*: $\rho(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$

Lasso:

- Recovery guarantees under various conditions (Incoherence, RIP, Restricted Eigenvalue, ...)
- Fast optimization schemes developed
- May yield suboptimal solutions

Exact / Approximate Mixed Integer Programming

- During optimization, calculate lower bound for objective
- If current objective equals lower bound, terminate with a global optimality certificate.

[Bertsimas, King, Mazumder, AoS '16]

Exact / Approximate Mixed Integer Programming

- During optimization, calculate lower bound for objective
- If current objective equals lower bound, terminate with a global optimality certificate.

  [Bertsimas, King, Mazumder, AoS '16]

- MIP solves (P0) *globally*
- Applicable with $d = O(100)$, much faster than exhaustive search

Exact / Approximate Mixed Integer Programming

- During optimization, calculate lower bound for objective
- If current objective equals lower bound, terminate with a global optimality certificate.

[Bertsimas, King, Mazumder, AoS '16]

- MIP solves (P0) *globally*
- Applicable with $d = O(100)$, much faster than exhaustive search

Limitation: May be very slow

Exact / Approximate Mixed Integer Programming

- During optimization, calculate lower bound for objective
- If current objective equals lower bound, terminate with a global optimality certificate.

[Bertsimas, King, Mazumder, AoS '16]

- MIP solves (P0) *globally*
- Applicable with $d = O(100)$, much faster than exhaustive search

Limitation: May be very slow

- On 30×180 matrix A and $k = 15$, may take several days

Exact / Approximate Mixed Integer Programming

- During optimization, calculate lower bound for objective
- If current objective equals lower bound, terminate with a global optimality certificate.

[Bertsimas, King, Mazumder, AoS '16]

- MIP solves (P0) *globally*
- Applicable with $d = O(100)$, much faster than exhaustive search

Limitation: May be very slow

- On 30×180 matrix A and $k = 15$, may take several days

[Bertsimas, Van Parys, AoS '20]

Cutting plane method

globally solve $d = 15000$, $n = 200$, $k = 10$ in minutes

Approximate MIP

[Hazimeh & Mazumder, Oper. Res. '20]

Greedy coordinate descent + local combinatorial search

Approximate MIP

[Hazimeh & Mazumder, Oper. Res. '20]

Greedy coordinate descent + local combinatorial search

- No optimality certificate
- Extremely fast, can handle $d = 10^6$ in less than a minute
- state of the art performance

Theoretical Guarantees

In addition to algorithm development, substantial body of literature on conditions for perfect recovery (noiseless setting), accurate and stable recovery in presence of noise.

Theoretical Guarantees

In addition to algorithm development, substantial body of literature on conditions for perfect recovery (noiseless setting), accurate and stable recovery in presence of noise.

Key notions: Coherence of dictionary, restricted isometry property, etc.

Theoretical Guarantees

In addition to algorithm development, substantial body of literature on conditions for perfect recovery (noiseless setting), accurate and stable recovery in presence of noise.

Key notions: Coherence of dictionary, restricted isometry property, etc.

Under some conditions, current methods are *optimal*

Theoretical Guarantees

In addition to algorithm development, substantial body of literature on conditions for perfect recovery (noiseless setting), accurate and stable recovery in presence of noise.

Key notions: Coherence of dictionary, restricted isometry property, etc.

Under some conditions, current methods are *optimal*

Has the problem not been solved yet?

Theoretical Guarantees

In addition to algorithm development, substantial body of literature on conditions for perfect recovery (noiseless setting), accurate and stable recovery in presence of noise.

Key notions: Coherence of dictionary, restricted isometry property, etc.

Under some conditions, current methods are *optimal*

Has the problem not been solved yet?

No !

Theoretical Guarantees

In addition to algorithm development, substantial body of literature on conditions for perfect recovery (noiseless setting), accurate and stable recovery in presence of noise.

Key notions: Coherence of dictionary, restricted isometry property, etc.

Under some conditions, current methods are *optimal*

Has the problem not been solved yet?

No !

Key limitation of above methods:
with few observations $n \ll d$,
higher values of k (not so sparse vectors)
nearly all prior methods either compute far from optimal solutions or run essentially forever...

Example

Matrix A of size 100×800, random i.i.d. $\mathcal{N}(0,1)$ entries followed by column normalization.

Example

Matrix A of size 100×800, random i.i.d. $\mathcal{N}(0, 1)$ entries followed by column normalization.

For various sparsity values k, generate random k-sparse vector $\boldsymbol{x}_0$. Its non-zero entries are i.i.d. $\mathcal{N}(0, 1)$.

Example

Matrix A of size 100×800, random i.i.d. $\mathcal{N}(0,1)$ entries followed by column normalization.

For various sparsity values k, generate random k-sparse vector $\boldsymbol{x}_0$. Its non-zero entries are i.i.d. $\mathcal{N}(0,1)$.

Generate

$$\boldsymbol{y} = A\boldsymbol{x}_0 + \mathbf{e}$$

where vector $\mathbf{e} \sim \sigma\mathcal{N}(\mathbf{0}, I_n)$, with $\mathbb{E}\|\mathbf{e}\|^2 = (0.05)^2 \cdot \mathbb{E}\|A\boldsymbol{x}_0\|^2$.

Example

Matrix A of size 100×800, random i.i.d. $\mathcal{N}(0,1)$ entries followed by column normalization.

For various sparsity values k, generate random k-sparse vector $\boldsymbol{x}_0$. Its non-zero entries are i.i.d. $\mathcal{N}(0,1)$.

Generate

$$\boldsymbol{y} = A\boldsymbol{x}_0 + \mathbf{e}$$

where vector $\mathbf{e} \sim \sigma\mathcal{N}(\mathbf{0}, I_n)$, with $\mathbb{E}\|\mathbf{e}\|^2 = (0.05)^2 \cdot \mathbb{E}\|A\boldsymbol{x}_0\|^2$.

Measure of optimization success:

$$\frac{\|A\hat{\boldsymbol{x}} - \boldsymbol{y}\|}{\|A\boldsymbol{x}_0 - \boldsymbol{y}\|}.$$

Example

Matrix A of size 100×800, random i.i.d. $\mathcal{N}(0,1)$ entries followed by column normalization.

For various sparsity values k, generate random k-sparse vector $\boldsymbol{x}_0$. Its non-zero entries are i.i.d. $\mathcal{N}(0,1)$.

Generate

$$\boldsymbol{y} = A\boldsymbol{x}_0 + \mathbf{e}$$

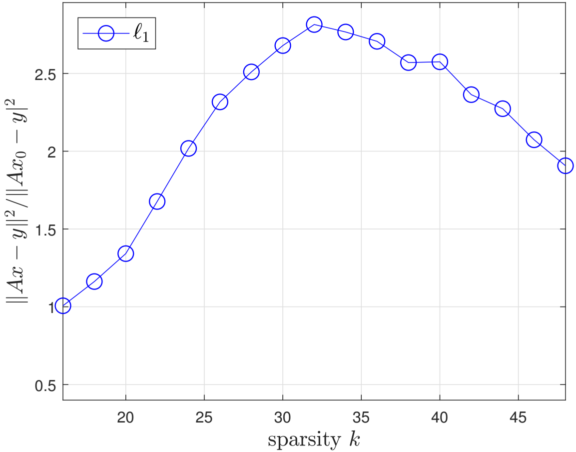
where vector $\mathbf{e} \sim \sigma\mathcal{N}(\mathbf{0}, I_n)$, with $\mathbb{E}\|\mathbf{e}\|^2 = (0.05)^2 \cdot \mathbb{E}\|A\boldsymbol{x}_0\|^2$.

Measure of optimization success:

$$\frac{\|A\hat{\boldsymbol{x}} - \boldsymbol{y}\|}{\|A\boldsymbol{x}_0 - \boldsymbol{y}\|}.$$

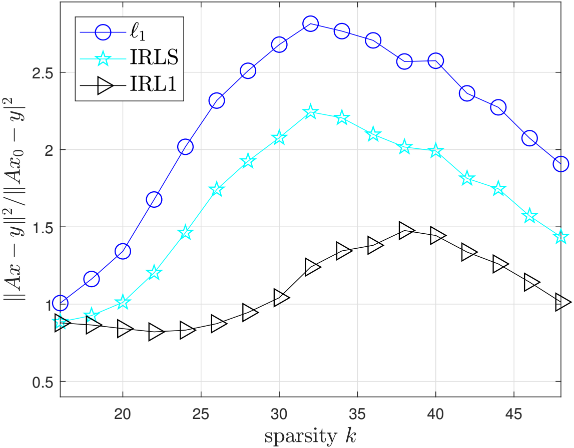
If ratio ≤ 1 then $\hat{\boldsymbol{x}}$ is *potentially* accurate estimate of $\boldsymbol{x}_0$

An Example



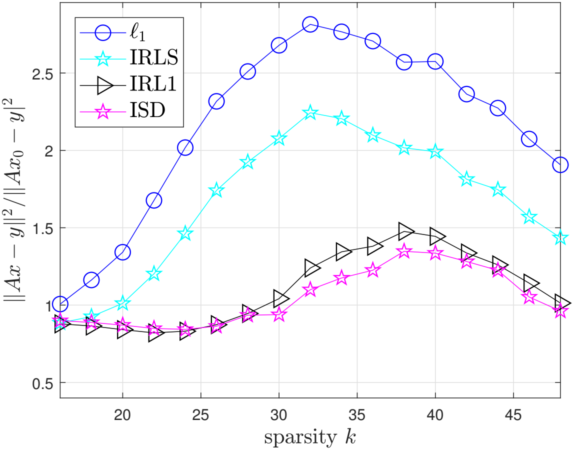
In our setting, ℓ_1 penalty (Lasso / Basis Pursuit) essentially works only up to sparsity levels $k \leq 16$.

An Example



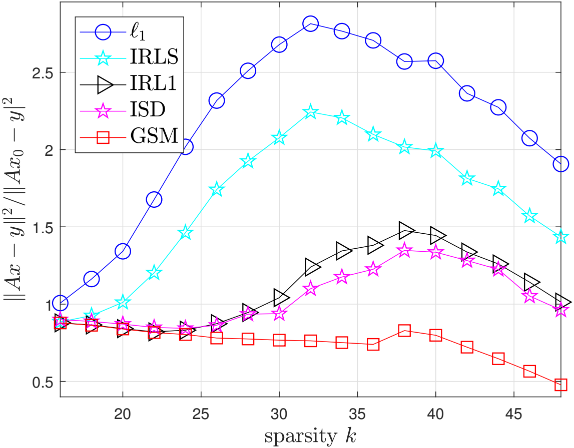
IRLS and IRL-1 solve ℓ_q penalized objectives with $q < 1$. Solved with 10 values of $q < 1$ and took solution with minimal $\|A\mathbf{x} - \mathbf{y}\|$.

An Example



ISD=Iterative Support Detection [Wang & Yin 2010'].
Sophisticated greedy support-detection strategy.

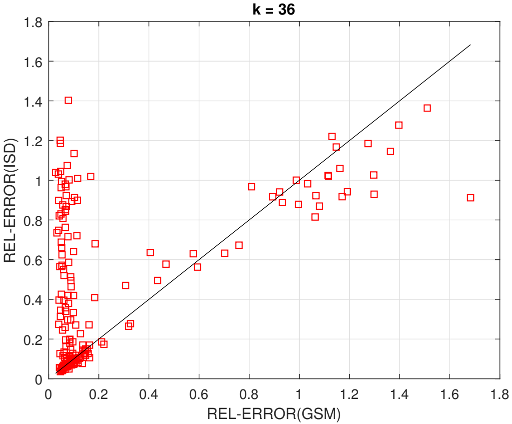
An Example



GSM= our proposed method. Superior at the more challenging settings with larger values of k and/or correlated dictionaries

An Example

Successful optimization often (but not always) translates into better recovery



Showing $\|\hat{\boldsymbol{x}} - \boldsymbol{x}_0\|_1 / \|\boldsymbol{x}_0\|_1$

Solving (P0) by a Penalized Objective

Desired properties for a penalty function:

Solving (P0) by a Penalized Objective

Desired properties for a penalty function:

(i) A penalty $\rho(\boldsymbol{x}) = \rho_k(\boldsymbol{x})$ that *explicitly* takes into account the sparsity level k

Solving (P0) by a Penalized Objective

Desired properties for a penalty function:

(i) A penalty $\rho(\boldsymbol{x}) = \rho_k(\boldsymbol{x})$ that *explicitly* takes into account the sparsity level k

(ii) For large λ, solutions of

$$\min \|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \rho_k(\boldsymbol{x})$$

are close to those of (P0).

- Better yet - they *coincide*

Solving (P0) by a Penalized Objective

Desired properties for a penalty function:

(i) A penalty $\rho(\boldsymbol{x}) = \rho_k(\boldsymbol{x})$ that *explicitly* takes into account the sparsity level k

(ii) For large λ, solutions of

$$\min \|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \rho_k(\boldsymbol{x})$$

are close to those of (P0).

- Better yet - they *coincide*

(iii) Objective would be easy to optimize

The Trimmed Lasso

A penalty that satisfies (i) and (ii) above: (Not our contribution)

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

where $|x|_{(1)} \geq |x|_{(2)} \geq \ldots \geq |x|_{(d)}$ are the entries of $\boldsymbol{x}$ in absolute value, sorted in decreasing order

The Trimmed Lasso

A penalty that satisfies (i) and (ii) above: (Not our contribution)

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

where $|x|_{(1)} \geq |x|_{(2)} \geq \ldots \geq |x|_{(d)}$ are the entries of $\boldsymbol{x}$ in absolute value, sorted in decreasing order

Penalize "tail" of $\boldsymbol{x}$: the ℓ_1 distance to the nearest k-sparse vector

The Trimmed Lasso

A penalty that satisfies (i) and (ii) above: (Not our contribution)

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

where $|x|_{(1)} \geq |x|_{(2)} \geq \ldots \geq |x|_{(d)}$ are the entries of $\boldsymbol{x}$ in absolute value, sorted in decreasing order

Penalize "tail" of $\boldsymbol{x}$: the ℓ_1 distance to the nearest k-sparse vector

Early related works:

- [Cohen, Dahmen, DeVore, *JAMS* '08]
- [Huang, Liu, Shi, Van Huffel, Suykens, *Sig. Proc.* '15]

The Trimmed Lasso

A penalty that satisfies (i) and (ii) above: (Not our contribution)

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

where $|x|_{(1)} \geq |x|_{(2)} \geq \ldots \geq |x|_{(d)}$ are the entries of $\boldsymbol{x}$ in absolute value, sorted in decreasing order

Penalize "tail" of $\boldsymbol{x}$: the ℓ_1 distance to the nearest k-sparse vector

Early related works:

- [Cohen, Dahmen, DeVore, *JAMS* '08]
- [Huang, Liu, Shi, Van Huffel, Suykens, *Sig. Proc.* '15]

Penalty studied by:

- [Gotoh, Takeda, Tono, *Math. Prog.* '18]
- [Bertsimas, Copenhaver, Mazumder, '17], who coined the term *trimmed Lasso*

The Trimmed Lasso

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Theoretical questions:

The Trimmed Lasso

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Theoretical questions:

1. Relation to original problem (P0)?

The Trimmed Lasso

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Theoretical questions:

1. Relation to original problem (P0)?
2. What value to use for λ?

The Trimmed Lasso

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Theoretical questions:

1. Relation to original problem (P0)?
2. What value to use for λ?
3. Can we recover $\boldsymbol{x}$ using $\tau_k(\boldsymbol{x})$?

The Trimmed Lasso

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Theoretical questions:

1. Relation to original problem (P0)?
2. What value to use for λ?
3. Can we recover $\boldsymbol{x}$ using $\tau_k(\boldsymbol{x})$?

Practical question: How to optimize an objective with $\tau_k(\boldsymbol{x})$?

The Trimmed Lasso

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Theoretical questions:

1. Relation to original problem (P0)?
2. What value to use for λ?
3. Can we recover $\boldsymbol{x}$ using $\tau_k(\boldsymbol{x})$?

Practical question: How to optimize an objective with $\tau_k(\boldsymbol{x})$?

Our contribution:

The Trimmed Lasso

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Theoretical questions:

1. Relation to original problem (P0)?
2. What value to use for λ?
3. Can we recover $\boldsymbol{x}$ using $\tau_k(\boldsymbol{x})$?

Practical question: How to optimize an objective with $\tau_k(\boldsymbol{x})$?

Our contribution:

1. Theoretical study of $\tau_k(\boldsymbol{x})$, addressing questions 1-3

The Trimmed Lasso

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Theoretical questions:

1. Relation to original problem (P0)?
2. What value to use for λ?
3. Can we recover $\boldsymbol{x}$ using $\tau_k(\boldsymbol{x})$?

Practical question: How to optimize an objective with $\tau_k(\boldsymbol{x})$?

Our contribution:

1. Theoretical study of $\tau_k(\boldsymbol{x})$, addressing questions 1-3
   $\rightarrow$ $\tau_k(\boldsymbol{x})$ is a good candidate for solving (P0)

The Trimmed Lasso

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Theoretical questions:

1. Relation to original problem (P0)?
2. What value to use for λ?
3. Can we recover $\boldsymbol{x}$ using $\tau_k(\boldsymbol{x})$?

Practical question: How to optimize an objective with $\tau_k(\boldsymbol{x})$?

Our contribution:

1. Theoretical study of $\tau_k(\boldsymbol{x})$, addressing questions 1-3
   $\rightarrow$ $\tau_k(\boldsymbol{x})$ is a good candidate for solving (P0)
2. Novel surrogate penalty that satisfies (i)-(iii)

The Trimmed Lasso

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Theoretical questions:

1. Relation to original problem (P0)?
2. What value to use for λ?
3. Can we recover $\boldsymbol{x}$ using $\tau_k(\boldsymbol{x})$?

Practical question: How to optimize an objective with $\tau_k(\boldsymbol{x})$?

Our contribution:

1. Theoretical study of $\tau_k(\boldsymbol{x})$, addressing questions 1-3
   $\rightarrow$ $\tau_k(\boldsymbol{x})$ is a good candidate for solving (P0)
2. Novel surrogate penalty that satisfies (i)-(iii)
3. Practical optimization method, state-of-the-art results

The Trimmed Lasso: Choosing λ

$$\min_{\boldsymbol{x}} \; F_\lambda(\boldsymbol{x}) := \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_k(\boldsymbol{x}) \qquad (P_\lambda)$$

The Trimmed Lasso: Choosing λ

$$\min_{\boldsymbol{x}} \ \mathrm{F}_\lambda(\boldsymbol{x}) := \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_k(\boldsymbol{x}) \qquad (\mathrm{P}_\lambda)$$

How to choose λ?

The Trimmed Lasso: Choosing λ

Define $\beta = \max_{i=1,\dots,d} \|\boldsymbol{a}_i\|_2$, where $\boldsymbol{a}_i$ are the columns of A.

Lemma

If $\lambda > \bar{\lambda} = \beta \|\boldsymbol{y}\|_2$, then any local minimum of (P_λ) is k-sparse.

The Trimmed Lasso: Choosing λ

Define $\beta = \max_{i=1,\dots,d} \|\boldsymbol{a}_i\|_2$, where $\boldsymbol{a}_i$ are the columns of A.

Lemma

If $\lambda > \bar{\lambda} = \beta\|\boldsymbol{y}\|_2$, then any local minimum of (P_λ) is k-sparse.

The Trimmed Lasso: Choosing λ

Define $\beta = \max_{i=1,\dots,d} \|\boldsymbol{a}_i\|_2$, where $\boldsymbol{a}_i$ are the columns of A.

Lemma

If $\lambda > \bar{\lambda} = \beta\|\boldsymbol{y}\|_2$, then any local minimum of (P_λ) is k-sparse.

- For large enough λ, optimal solutions of (P_λ) coincide with those of (P0).

The Trimmed Lasso: Choosing λ

Define $\beta = \max_{i=1,\ldots,d} \|\boldsymbol{a}_i\|_2$, where $\boldsymbol{a}_i$ are the columns of A.

Lemma

If $\lambda > \bar{\lambda} = \beta\|\boldsymbol{y}\|_2$, then any local minimum of (P_λ) is k-sparse.

- For large enough λ, optimal solutions of (P_λ) coincide with those of (P0).
- Strategy: Solve with increasing values of λ, until a k-sparse solution is obtained.
  - $\rightarrow$ Guaranteed to happen when λ surpasses the threshold.

Sparse Signal Recovery Guarantees

Suppose that

$$\boldsymbol{y} = A\boldsymbol{x}_0 + \boldsymbol{e} \in \mathbb{R}^n$$

$\boldsymbol{x}_0 \in \mathbb{R}^d$ = unknown vector to be recovered
$\boldsymbol{e}$ = measurement error

Sparse Signal Recovery Guarantees

Suppose that

$$\boldsymbol{y} = A\boldsymbol{x}_0 + \boldsymbol{e} \in \mathbb{R}^n$$

$\boldsymbol{x}_0 \in \mathbb{R}^d$ = unknown vector to be recovered

$\boldsymbol{e}$ = measurement error

Assumptions:

- $\boldsymbol{x}_0$ is approximately k-sparse ($\tau_k(\boldsymbol{x}_0) \ll \|\boldsymbol{x}_0\|_1$)
- $\|\boldsymbol{e}\|_2$ is small

Sparse Signal Recovery Guarantees

Suppose that

$$\boldsymbol{y} = A\boldsymbol{x}_0 + \boldsymbol{e} \in \mathbb{R}^n$$

$\boldsymbol{x}_0 \in \mathbb{R}^d$ = unknown vector to be recovered

$\boldsymbol{e}$ = measurement error

Assumptions:

- $\boldsymbol{x}_0$ is approximately k-sparse ($\tau_k(\boldsymbol{x}_0) \ll \|\boldsymbol{x}_0\|_1$)
- $\|\boldsymbol{e}\|_2$ is small

Goal: Recover $\boldsymbol{x}_0$ given $A, \boldsymbol{y}$ and k.

Sparse Signal Recovery Guarantees

Suppose that

$$\boldsymbol{y} = A\boldsymbol{x}_0 + \boldsymbol{e} \in \mathbb{R}^n$$

$\boldsymbol{x}_0 \in \mathbb{R}^d$ = unknown vector to be recovered
$\boldsymbol{e}$ = measurement error

Assumptions:

- $\boldsymbol{x}_0$ is approximately k-sparse $(\tau_k(\boldsymbol{x}_0) \ll \|\boldsymbol{x}_0\|_1)$
- $\|\boldsymbol{e}\|_2$ is small

Goal: Recover $\boldsymbol{x}_0$ given $A, \boldsymbol{y}$ and k.

Question:

Can one accurately recover $\boldsymbol{x}_0$ by solving problem (P_λ) ?

Sparse Signal Recovery

Without additional assumptions on A, this problem is ill posed

Sparse Signal Recovery

Without additional assumptions on A, this problem is ill posed

- Even in the absence of noise, to be able to recover $\boldsymbol{x}_0$, any $2k$ columns of A must be linearly independent

Sparse Signal Recovery

Without additional assumptions on A, this problem is ill posed

- Even in the absence of noise, to be able to recover $\boldsymbol{x}_0$, any $2k$ columns of A must be linearly independent

Assumption

There exists a constant $\alpha_{2k} > 0$ *such that for all* $\boldsymbol{x} \in \mathbb{R}^d$ *with* $\|\boldsymbol{x}\|_0 \leq 2k$,

$$\|A\boldsymbol{x}\|_2 \geq \alpha_{2k}\|\boldsymbol{x}\|_1$$

Sparse Signal Recovery

Without additional assumptions on A, this problem is ill posed

- Even in the absence of noise, to be able to recover $\boldsymbol{x}_0$, any $2k$ columns of A must be linearly independent

Assumption

There exists a constant $\alpha_{2k} > 0$ *such that for all* $\boldsymbol{x} \in \mathbb{R}^d$ *with* $\|\boldsymbol{x}\|_0 \leq 2k$,

$$\|A\boldsymbol{x}\|_2 \geq \alpha_{2k}\|\boldsymbol{x}\|_1$$

Variant of the *Restricted Isometry Property*: One-sided, with mixed norms

Sparse Signal Recovery

Without additional assumptions on A, this problem is ill posed

- Even in the absence of noise, to be able to recover $\boldsymbol{x}_0$, any $2k$ columns of A must be linearly independent

Assumption

There exists a constant $\alpha_{2k} > 0$ *such that for all* $\boldsymbol{x} \in \mathbb{R}^d$ *with* $\|\boldsymbol{x}\|_0 \leq 2k$,

$$\|A\boldsymbol{x}\|_2 \geq \alpha_{2k}\|\boldsymbol{x}\|_1$$

Variant of the *Restricted Isometry Property*: One-sided, with mixed norms

Notation:

For a vector $\boldsymbol{x} \in \mathbb{R}^d$, denote by $\Pi_k(\boldsymbol{x})$ the k-sparse *projection* of $\boldsymbol{x}$, namely the nearest k-sparse vector to $\boldsymbol{x}$

The Trimmed Lasso: Sparse Recovery Guarantees

Theorem

Suppose that for some $\lambda > 0$, an optimization algorithm outputs a solution $\hat{\boldsymbol{x}}$ such that

$$F_\lambda(\hat{\boldsymbol{x}}) \leq F_\lambda(\Pi_k(\boldsymbol{x}_0)).$$

The Trimmed Lasso: Sparse Recovery Guarantees

Theorem

Suppose that for some $\lambda > 0$, an optimization algorithm outputs a solution $\hat{\boldsymbol{x}}$ such that

$$F_\lambda(\hat{\boldsymbol{x}}) \leq F_\lambda(\Pi_k(\boldsymbol{x}_0)).$$

Let $\xi = \|\mathbf{e}\|_2 + \beta\tau_k(\boldsymbol{x}_0)$. Then,

The Trimmed Lasso: Sparse Recovery Guarantees

Theorem

Suppose that for some $\lambda > 0$, an optimization algorithm outputs a solution $\hat{\boldsymbol{x}}$ such that

$$F_\lambda(\hat{\boldsymbol{x}}) \leq F_\lambda(\Pi_k(\boldsymbol{x}_0)).$$

Let $\xi = \|\mathbf{e}\|_2 + \beta\tau_k(\boldsymbol{x}_0)$. Then,

1. *The projected solution $\Pi_k(\hat{\boldsymbol{x}})$ is close to $\boldsymbol{x}_0$,*

$$\|\Pi_k(\hat{\boldsymbol{x}}) - \boldsymbol{x}_0\|_1 \leq \tau_k(\boldsymbol{x}_0) + \frac{2}{\alpha_{2k}}\xi + \frac{1}{2\lambda\alpha_{2k}}\xi^2$$

The Trimmed Lasso: Sparse Recovery Guarantees

Theorem

Suppose that for some $\lambda > 0$, an optimization algorithm outputs a solution $\hat{\boldsymbol{x}}$ such that

$$F_\lambda(\hat{\boldsymbol{x}}) \leq F_\lambda(\Pi_k(\boldsymbol{x}_0)).$$

Let $\xi = \|\mathbf{e}\|_2 + \beta\tau_k(\boldsymbol{x}_0)$. Then,

1. *The projected solution $\Pi_k(\hat{\boldsymbol{x}})$ is close to $\boldsymbol{x}_0$,*

$$\|\Pi_k(\hat{\boldsymbol{x}}) - \boldsymbol{x}_0\|_1 \leq \tau_k(\boldsymbol{x}_0) + \frac{2}{\alpha_{2k}}\xi + \frac{1}{2\lambda\alpha_{2k}}\xi^2$$

2. *If $\hat{\boldsymbol{x}}$ itself is k-sparse, then the following tighter bound holds,*

$$\|\hat{\boldsymbol{x}} - \boldsymbol{x}_0\|_1 \leq \tau_k(\boldsymbol{x}_0) + \frac{2}{\alpha_{2k}}\xi$$

The Trimmed Lasso: Sparse Recovery Guarantees

Implication: We can well-approximate $\boldsymbol{x}_0$ by solving (P_λ) with λ *smaller* than $\bar{\lambda}$

The Trimmed Lasso: Sparse Recovery Guarantees

Implication: We can well-approximate $\boldsymbol{x}_0$ by solving (P_λ) with λ *smaller* than $\bar{\lambda}$

- We don't need the optimal solutions of (P_λ) to coincide with those of (P0)

The Trimmed Lasso: Sparse Recovery Guarantees

Implication: We can well-approximate $\boldsymbol{x}_0$ by solving (P_λ) with λ *smaller* than $\bar{\lambda}$

- We don't need the optimal solutions of (P_λ) to coincide with those of (P0)
- Potentially, solving (P_λ) with smaller λ is easier

Implication: We can well-approximate $\boldsymbol{x}_0$ by solving (P_λ) with λ *smaller* than $\bar{\lambda}$

- We don't need the optimal solutions of (P_λ) to coincide with those of (P0)
- Potentially, solving (P_λ) with smaller λ is easier
- Recovery is stable w.r.t. measurement error $\|\mathbf{e}\|_2$ and inexactness of sparsity $\tau_k(\boldsymbol{x}_0)$

Note: Theoretical guarantee for Lasso has better dependence on $\tau_k(\boldsymbol{x}_0)$, by a factor of $\mathcal{O}\left(\sqrt{k}\right)$.

The Trimmed Lasso: Sparse Recovery Guarantees

Note: Theoretical guarantee for Lasso has better dependence on $\tau_k(\boldsymbol{x}_0)$, by a factor of $\mathcal{O}\left(\sqrt{k}\right)$.

- However, it requires the RIP constant to be bounded away from zero.

The Trimmed Lasso: Sparse Recovery Guarantees

Note: Theoretical guarantee for Lasso has better dependence on $\tau_k(\boldsymbol{x}_0)$, by a factor of $\mathcal{O}\left(\sqrt{k}\right)$.

- However, it requires the RIP constant to be bounded away from zero.
  Even w/out noise, Lasso/BP requires α_{2k} to be bounded away from zero for recovery guarantees.

Note: Theoretical guarantee for Lasso has better dependence on $\tau_k(\boldsymbol{x}_0)$, by a factor of $\mathcal{O}\left(\sqrt{k}\right)$.

- However, it requires the RIP constant to be bounded away from zero.
  Even w/out noise, Lasso/BP requires α_{2k} to be bounded away from zero for recovery guarantees.
- Our guarantee only requires $\alpha_{2k} > 0$.

The Trimmed Lasso: Sparse Recovery Guarantees

Note: Theoretical guarantee for Lasso has better dependence on $\tau_k(\boldsymbol{x}_0)$, by a factor of $\mathcal{O}\left(\sqrt{k}\right)$.

- However, it requires the RIP constant to be bounded away from zero.
  Even w/out noise, Lasso/BP requires α_{2k} to be bounded away from zero for recovery guarantees.
- Our guarantee only requires $\alpha_{2k} > 0$.
  - $\rightarrow$ a necessary condition for successful recovery by *any* algorithm

The Trimmed Lasso: Sparse Recovery Guarantees

Note: Theoretical guarantee for Lasso has better dependence on $\tau_k(\boldsymbol{x}_0)$, by a factor of $\mathcal{O}\left(\sqrt{k}\right)$.

- However, it requires the RIP constant to be bounded away from zero.
  Even w/out noise, Lasso/BP requires α_{2k} to be bounded away from zero for recovery guarantees.
- Our guarantee only requires $\alpha_{2k} > 0$.
  - $\rightarrow$ a necessary condition for successful recovery by *any* algorithm

In conclusion:

Optimizing trimmed-lasso penalized objectives is a promising approach to (P0).

The Trimmed Lasso: Practical Optimization

Reminder:

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

The Trimmed Lasso: Practical Optimization

Reminder:

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Goal:

$$\boxed{\min_{\boldsymbol{x}} \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_k(\boldsymbol{x})}$$

The Trimmed Lasso: Practical Optimization

Reminder:

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Goal:

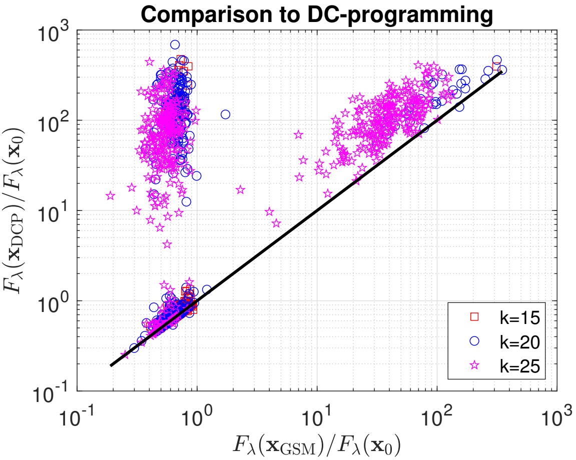
$$\boxed{\min_{\boldsymbol{x}} \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_k(\boldsymbol{x})}$$

Previous Optimization Methods:

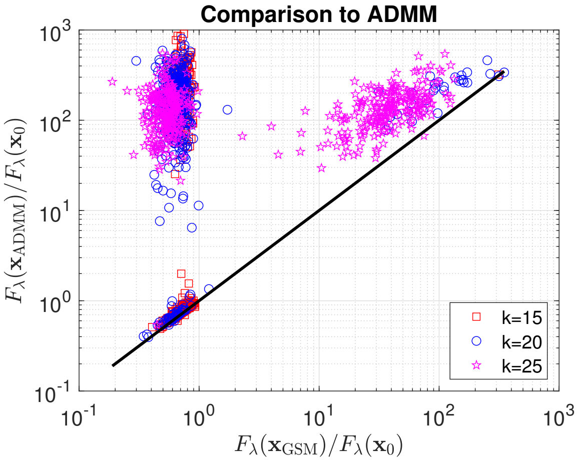
◦ Difference of Convex Programming (DCP)
[Gotoh, Takeda, Tono, *Math. Prog.* '18]

◦ Alternating Direction Method of Multipliers (ADMM)
[Bertsimas, Copenhaver, Mazumder, '17]

The Trimmed Lasso: Practical Optimization



The Trimmed Lasso: Practical Optimization



The Trimmed Lasso: Practical Optimization

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

The Trimmed Lasso: Practical Optimization

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Alternative formula:

$$\tau_k(\boldsymbol{x}) = \min_{|\Lambda|=d-k} \sum_{i\in\Lambda} |x_i|$$

The Trimmed Lasso: Practical Optimization

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Alternative formula:

$$\tau_k(\boldsymbol{x}) = \min_{|\Lambda|=d-k} \sum_{i\in\Lambda} |x_i|$$

Trimmed Lasso as a *hard* minimum:
Out of all $\binom{d}{k}$ subsets of $\{1, \ldots, d\}$, choose one with minimal ℓ_1-norm.

The Trimmed Lasso: Practical Optimization

$$\tau_k(\boldsymbol{x}) = \sum_{j=k+1}^{d} |x|_{(j)}$$

Alternative formula:

$$\tau_k(\boldsymbol{x}) = \min_{|\Lambda|=d-k} \sum_{i\in\Lambda} |x_i|$$

Trimmed Lasso as a *hard* minimum:

Out of all $\binom{d}{k}$ subsets of $\{1,\ldots,d\}$, choose one with minimal ℓ_1-norm.

Our Key Idea: Replace the hard minimum by a *soft* minimum.

Surrogate for Trimmed Lasso

Let $\boldsymbol{z} \in \mathbb{R}^m$ with $m = \binom{d}{k}$, whose entries consist of the ℓ_1-norms of all subvectors of $\boldsymbol{x}$ of size $d - k$. Formally:

Surrogate for Trimmed Lasso

Let $\boldsymbol{z} \in \mathbb{R}^m$ with $m = \binom{d}{k}$, whose entries consist of the ℓ_1-norms of all subvectors of $\boldsymbol{x}$ of size $d - k$. Formally:

$\boldsymbol{z}$ is indexed by subsets $\Lambda \subset \{1, \ldots, d\}$ of size $d - k$:

$$\boldsymbol{z} = (\boldsymbol{z}_\Lambda), \quad |\Lambda| = d - k$$

Surrogate for Trimmed Lasso

Let $\boldsymbol{z} \in \mathbb{R}^m$ with $m = \binom{d}{k}$, whose entries consist of the ℓ_1-norms of all subvectors of $\boldsymbol{x}$ of size $d-k$. Formally:

$\boldsymbol{z}$ is indexed by subsets $\Lambda \subset \{1, \ldots, d\}$ of size $d-k$:

$$\boldsymbol{z} = (\boldsymbol{z}_\Lambda), \quad |\Lambda| = d - k$$

Each entry of $\boldsymbol{z}$ is given by

$$z_\Lambda = \sum_{i \in \Lambda} |x_i|$$

Surrogate for Trimmed Lasso

Let $\boldsymbol{z} \in \mathbb{R}^m$ with $m = \binom{d}{k}$, whose entries consist of the ℓ_1-norms of all subvectors of $\boldsymbol{x}$ of size $d-k$. Formally:

$\boldsymbol{z}$ is indexed by subsets $\Lambda \subset \{1, \ldots, d\}$ of size $d-k$:

$$\boldsymbol{z} = (\boldsymbol{z}_\Lambda), \quad |\Lambda| = d - k$$

Each entry of $\boldsymbol{z}$ is given by

$$z_\Lambda = \sum_{i \in \Lambda} |x_i|$$

Note that

$$\tau_k(\boldsymbol{x}) = \min_{|\Lambda| = d-k} \; z_\Lambda$$

Surrogate for Trimmed Lasso

Let $\boldsymbol{z} \in \mathbb{R}^m$ with $m = \binom{d}{k}$, whose entries consist of the ℓ_1-norms of all subvectors of $\boldsymbol{x}$ of size $d - k$. Formally:

$\boldsymbol{z}$ is indexed by subsets $\Lambda \subset \{1, \ldots, d\}$ of size $d - k$:

$$\boldsymbol{z} = (\boldsymbol{z}_\Lambda), \quad |\Lambda| = d - k$$

Each entry of $\boldsymbol{z}$ is given by

$$z_\Lambda = \sum_{i \in \Lambda} |x_i|$$

We wish:

$$\rho(\boldsymbol{x}) = \underset{|\Lambda| = d-k}{\text{soft min}} \; z_\Lambda$$

Surrogate for Trimmed Lasso

Let $\boldsymbol{z} \in \mathbb{R}^m$ with $m = \binom{d}{k}$, whose entries consist of the ℓ_1-norms of all subvectors of $\boldsymbol{x}$ of size $d-k$. Formally:

$\boldsymbol{z}$ is indexed by subsets $\Lambda \subset \{1,\ldots,d\}$ of size $d-k$:

$$\boldsymbol{z} = (\boldsymbol{z}_\Lambda), \quad |\Lambda| = d-k$$

Each entry of $\boldsymbol{z}$ is given by

$$z_\Lambda = \sum_{i\in\Lambda} |x_i|$$

We wish:

$$\rho(\boldsymbol{x}) = \underset{|\Lambda|=d-k}{\text{soft min}}\ z_\Lambda$$

- As in the *softmax* function in multi-class classification.

Surrogate for Trimmed Lasso

Soft maximum of $\boldsymbol{z} = (z_1, \ldots, z_m)$:

$$\log \left(\sum_{j=1}^{m} \exp(z_j) \right)$$

Surrogate for Trimmed Lasso

Soft minimum of $\boldsymbol{z}$:

$$-\log\left(\sum_{j=1}^{m}\exp\left(-z_j\right)\right)$$

Surrogate for Trimmed Lasso

Add a smoothness parameter γ:

$$-\frac{1}{\gamma} \log \left(\sum_{j=1}^{m} \exp\left(-\gamma z_j\right) \right)$$

Surrogate for Trimmed Lasso

Add averaging:

$$-\frac{1}{\gamma}\log\left(\frac{1}{m}\sum_{j=1}^{m}\exp\left(-\gamma z_j\right)\right)$$

Surrogate for Trimmed Lasso

Plug in the original definition of $\boldsymbol{z}$:

$$-\frac{1}{\gamma}\log\left(\frac{1}{\binom{d}{k}}\sum_{|\Lambda|=d-k}\exp\left(-\gamma\sum_{i\in\Lambda}|x_i|\right)\right)$$

Surrogate for Trimmed Lasso

$$-\frac{1}{\gamma}\log\left(\frac{1}{\binom{d}{k}}\sum_{|\Lambda|=d-k}\exp\Big(-\gamma\sum_{i\in\Lambda}|x_i|\Big)\right)$$

Surrogate for Trimmed Lasso

$$\tau_{k,\gamma}(\boldsymbol{x}) = -\frac{1}{\gamma}\log\left(\frac{1}{\binom{d}{k}}\sum_{|\Lambda|=d-k}\exp\Big(-\gamma\sum_{i\in\Lambda}|x_i|\Big)\right)$$

Generalized Soft-Min Penalty

Surrogate for Trimmed Lasso

$$\tau_{k,\gamma}(\boldsymbol{x}) = -\frac{1}{\gamma}\log\left(\frac{1}{\binom{d}{k}}\sum_{|\Lambda|=d-k}\exp\left(-\gamma\sum_{i\in\Lambda}|x_i|\right)\right)$$

Generalized Soft-Min Penalty

- Infinitely differentiable as a function of $|\boldsymbol{x}|$

Surrogate for Trimmed Lasso

$$\tau_{k,\gamma}(\boldsymbol{x}) = -\frac{1}{\gamma}\log\left(\frac{1}{\binom{d}{k}}\sum_{|\Lambda|=d-k}\exp\left(-\gamma\sum_{i\in\Lambda}|x_i|\right)\right)$$

Generalized Soft-Min Penalty

- Infinitely differentiable as a function of $|\boldsymbol{x}|$
  - Parameter γ controls level of smoothness

Surrogate for Trimmed Lasso

$$\tau_{k,\gamma}(\boldsymbol{x}) = -\frac{1}{\gamma}\log\left(\frac{1}{\binom{d}{k}}\sum_{|\Lambda|=d-k}\exp\left(-\gamma\sum_{i\in\Lambda}|x_i|\right)\right)$$

Generalized Soft-Min Penalty

- Infinitely differentiable as a function of $|\boldsymbol{x}|$
  - Parameter γ controls level of smoothness
- Takes into account all possible $\binom{d}{k}$ sparsity patterns of $\boldsymbol{x}$

Surrogate for Trimmed Lasso

$$\tau_{k,\gamma}(\boldsymbol{x}) = -\frac{1}{\gamma}\log\left(\frac{1}{\binom{d}{k}}\sum_{|\Lambda|=d-k}\exp\Big(-\gamma\sum_{i\in\Lambda}|x_i|\Big)\right)$$

Generalized Soft-Min Penalty

- Infinitely differentiable as a function of $|\boldsymbol{x}|$
  - Parameter γ controls level of smoothness
- Takes into account all possible $\binom{d}{k}$ sparsity patterns of $\boldsymbol{x}$
- Significantly easier to optimize

Generalized Soft-Min Properties

Lemma

For any $\boldsymbol{x} \in \mathbb{R}^d$*, the function* $\tau_{k,\gamma}(\boldsymbol{x})$ *is monotone-decreasing with respect to* γ*. Moreover,*

Generalized Soft-Min Properties

Lemma

For any $\boldsymbol{x} \in \mathbb{R}^d$, *the function* $\tau_{k,\gamma}(\boldsymbol{x})$ *is monotone-decreasing with respect to* γ. *Moreover,*

$$\lim_{\gamma \to 0} \tau_{k,\gamma}(\boldsymbol{x}) = \frac{d-k}{d} \|\boldsymbol{x}\|_1$$

Generalized Soft-Min Properties

Lemma

For any $\boldsymbol{x} \in \mathbb{R}^d$, the function $\tau_{k,\gamma}(\boldsymbol{x})$ is monotone-decreasing with respect to γ. Moreover,

$$\lim_{\gamma \to 0} \tau_{k,\gamma}(\boldsymbol{x}) = \frac{d-k}{d} \|\boldsymbol{x}\|_1$$

$$\lim_{\gamma \to \infty} \tau_{k,\gamma}(\boldsymbol{x}) = \tau_k(\boldsymbol{x})$$

Generalized Soft-Min Properties

Lemma

For any $\boldsymbol{x} \in \mathbb{R}^d$, *the function* $\tau_{k,\gamma}(\boldsymbol{x})$ *is monotone-decreasing with respect to* γ. *Moreover,*

$$\lim_{\gamma \to 0} \tau_{k,\gamma}(\boldsymbol{x}) = \tfrac{d-k}{d} \|\boldsymbol{x}\|_1$$

$$\lim_{\gamma \to \infty} \tau_{k,\gamma}(\boldsymbol{x}) = \tau_k(\boldsymbol{x})$$

A Homotopy Scheme

Instead of directly minimizing

$$\frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_k(\boldsymbol{x})$$

A Homotopy Scheme

Instead of directly minimizing

$$\frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \tau_k(\boldsymbol{x})$$

Solve a sequence of problems

$$\min_{\boldsymbol{x}} \mathsf{F}_{\lambda,\gamma}(\boldsymbol{x}) = \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \tau_{k,\gamma}(\boldsymbol{x})$$

with an increasing sequence $\gamma_0 < \gamma_1 < \ldots$, while tracing path of solutions.

A Homotopy Scheme

Instead of directly minimizing

$$\frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_k(\boldsymbol{x})$$

Solve a sequence of problems

$$\min_{\boldsymbol{x}} \mathsf{F}_{\lambda,\gamma}(\boldsymbol{x}) = \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_{k,\gamma}(\boldsymbol{x})$$

with an increasing sequence $\gamma_0 < \gamma_1 < \ldots$, while tracing path of solutions.

- Start at $\gamma = 0$: $\tau_{k,0}(\boldsymbol{x})$ is the convex ℓ_1 norm (Lasso problem).

A Homotopy Scheme

Instead of directly minimizing

$$\frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_k(\boldsymbol{x})$$

Solve a sequence of problems

$$\min_{\boldsymbol{x}} \mathrm{F}_{\lambda,\gamma}(\boldsymbol{x}) = \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_{k,\gamma}(\boldsymbol{x})$$

with an increasing sequence $\gamma_0 < \gamma_1 < \ldots$, while tracing path of solutions.

- Start at $\gamma = 0$: $\tau_{k,0}(\boldsymbol{x})$ is the convex ℓ_1 norm (Lasso problem).
- Slowly increase γ. At iteration t with $\gamma = \gamma_t$, initialize optimization method with previous solution $\hat{\boldsymbol{x}}_{t-1}$.

Majorization Minimization Scheme

Problem: How to minimize each nonconvex objective

$$\mathsf{F}_{\lambda,\gamma}(\boldsymbol{x}) = \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_{k,\gamma}(\boldsymbol{x})?$$

Majorization Minimization Scheme

Problem: How to minimize each nonconvex objective

$$F_{\lambda,\gamma}(\boldsymbol{x}) = \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_{k,\gamma}(\boldsymbol{x})?$$

Approach: Majorization-Minimization

Majorization Minimization Scheme

Problem: How to minimize each nonconvex objective

$$\mathrm{F}_{\lambda,\gamma}(\boldsymbol{x}) = \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_{k,\gamma}(\boldsymbol{x})?$$

Approach: Majorization-Minimization

Construct a function $\mathrm{G}_{\lambda,\gamma}(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ such that

$$\mathrm{G}_{\lambda,\gamma}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \geq \mathrm{F}_{\lambda,\gamma}(\boldsymbol{x}), \quad \mathrm{G}_{\lambda,\gamma}(\boldsymbol{x}, \boldsymbol{x}) = \mathrm{F}_{\lambda,\gamma}(\boldsymbol{x}).$$

Iterate:

$$\boldsymbol{x}^t = \arg\min_x \mathrm{G}_{\lambda,\gamma}\left(\boldsymbol{x}, \boldsymbol{x}^{t-1}\right).$$

Majorization Minimization Scheme

Problem: How to minimize each nonconvex objective

$$\mathsf{F}_{\lambda,\gamma}(\boldsymbol{x}) = \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_{k,\gamma}(\boldsymbol{x})?$$

Approach: Majorization-Minimization

Construct a function $\mathsf{G}_{\lambda,\gamma}(\boldsymbol{x},\tilde{\boldsymbol{x}})$ such that

$$\mathsf{G}_{\lambda,\gamma}(\boldsymbol{x},\tilde{\boldsymbol{x}}) \geq \mathsf{F}_{\lambda,\gamma}(\boldsymbol{x}), \quad \mathsf{G}_{\lambda,\gamma}(\boldsymbol{x},\boldsymbol{x}) = \mathsf{F}_{\lambda,\gamma}(\boldsymbol{x}).$$

Iterate:

$$\boldsymbol{x}^t = \arg\min_x \mathsf{G}_{\lambda,\gamma}\left(\boldsymbol{x},\boldsymbol{x}^{t-1}\right).$$

- Objective is guaranteed to decrease monotonically.

Majorization Minimization Scheme

Problem: How to minimize each nonconvex objective

$$\mathsf{F}_{\lambda,\gamma}(\boldsymbol{x}) = \tfrac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_{k,\gamma}(\boldsymbol{x})?$$

Approach: Majorization-Minimization

Construct a function $\mathsf{G}_{\lambda,\gamma}(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ such that

$$\mathsf{G}_{\lambda,\gamma}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \geq \mathsf{F}_{\lambda,\gamma}(\boldsymbol{x}), \quad \mathsf{G}_{\lambda,\gamma}(\boldsymbol{x}, \boldsymbol{x}) = \mathsf{F}_{\lambda,\gamma}(\boldsymbol{x}).$$

Iterate:

$$\boldsymbol{x}^t = \arg\min_x \mathsf{G}_{\lambda,\gamma}\left(\boldsymbol{x}, \boldsymbol{x}^{t-1}\right).$$

- Objective is guaranteed to decrease monotonically.
- Under some assumptions, guaranteed to converge to a stationary point.

Majorization Minimization Scheme

Constructing a majorizer for $\mathbf{F}_{\lambda,\gamma}(\boldsymbol{x})$:

Majorization Minimization Scheme

Constructing a majorizer for $\mathbf{F}_{\lambda,\gamma}(\boldsymbol{x})$:

Define $w_{k,\gamma} : \mathbb{R}^d \to \mathbb{R}^d$ for $0 \leq \gamma < \infty$ by

$$w^i_{k,\gamma}(\boldsymbol{x}) = \frac{\sum_{|\Lambda|=d-k, i\in\Lambda} \exp\left(-\gamma \sum_{j\in\Lambda} |x_j|\right)}{\sum_{|\Lambda|=d-k} \exp\left(-\gamma \sum_{j\in\Lambda} |x_j|\right)}$$

Majorization Minimization Scheme

Constructing a majorizer for $\mathsf{F}_{\lambda,\gamma}(\boldsymbol{x})$:

Define $w_{k,\gamma} : \mathbb{R}^d \to \mathbb{R}^d$ for $0 \leq \gamma < \infty$ by

$$w^i_{k,\gamma}(\boldsymbol{x}) = \frac{\sum_{|\Lambda|=d-k, i\in\Lambda} \exp\left(-\gamma \sum_{j\in\Lambda} |x_j|\right)}{\sum_{|\Lambda|=d-k} \exp\left(-\gamma \sum_{j\in\Lambda} |x_j|\right)}$$

Lemma: The following function is a majorizer of $\mathsf{F}_{\lambda,\gamma}(\boldsymbol{x})$:

$$\mathsf{G}_{\lambda,\gamma}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\tau_{k,\gamma}(\tilde{\boldsymbol{x}}) + \lambda\langle w_{k,\gamma}(\tilde{\boldsymbol{x}}), |\boldsymbol{x}| - |\tilde{\boldsymbol{x}}|\rangle$$

Majorization Minimization Scheme

Constructing a majorizer for $\mathsf{F}_{\lambda,\gamma}(\boldsymbol{x})$:

Define $w_{k,\gamma} : \mathbb{R}^d \to \mathbb{R}^d$ for $0 \le \gamma < \infty$ by

$$w^i_{k,\gamma}(\boldsymbol{x}) = \frac{\sum_{|\Lambda|=d-k, i\in\Lambda} \exp\left(-\gamma \sum_{j\in\Lambda} |x_j|\right)}{\sum_{|\Lambda|=d-k} \exp\left(-\gamma \sum_{j\in\Lambda} |x_j|\right)}$$

Lemma: The following function is a majorizer of $\mathsf{F}_{\lambda,\gamma}(\boldsymbol{x})$:

$$\mathsf{G}_{\lambda,\gamma}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\tau_{k,\gamma}(\tilde{\boldsymbol{x}}) + \lambda\langle w_{k,\gamma}(\tilde{\boldsymbol{x}}), |\boldsymbol{x}| - |\tilde{\boldsymbol{x}}|\rangle$$

constant w.r.t. $\boldsymbol{x}$

Majorization Minimization Scheme

MM scheme to minimize $\mathbf{F}_{\lambda,\gamma}(\boldsymbol{x})$:

$$\mathbf{w}^t = \mathbf{w}_{k,\gamma}(\boldsymbol{x}^{t-1})$$
$$\boldsymbol{x}^t = \arg\min_x \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\langle \mathbf{w}^t, |\boldsymbol{x}|\rangle$$

Majorization Minimization Scheme

MM scheme to minimize $\mathbf{F}_{\lambda,\gamma}(\boldsymbol{x})$:

$$\mathbf{w}^t = \mathbf{w}_{k,\gamma}(\boldsymbol{x}^{t-1})$$
$$\boldsymbol{x}^t = \arg\min_x \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\langle \mathbf{w}^t, |\boldsymbol{x}|\rangle$$

Each subproblem is a *convex* weighted ℓ_1 problem.

Majorization Minimization Scheme

MM scheme to minimize $\mathbf{F}_{\lambda,\gamma}(\boldsymbol{x})$:

$$\mathbf{w}^t = \mathbf{w}_{k,\gamma}(\boldsymbol{x}^{t-1})$$

$$\boldsymbol{x}^t = \arg\min_x \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\langle \mathbf{w}^t, |\boldsymbol{x}|\rangle$$

Each subproblem is a *convex* weighted ℓ_1 problem.
Similar to IRL1...

Majorization Minimization Scheme

MM scheme to minimize $\mathbf{F}_{\lambda,\gamma}(\boldsymbol{x})$:

$$\mathbf{w}^t = \mathbf{w}_{k,\gamma}(\boldsymbol{x}^{t-1})$$
$$\boldsymbol{x}^t = \arg\min_x \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\langle \mathbf{w}^t, |\boldsymbol{x}|\rangle$$

Each subproblem is a *convex* weighted ℓ_1 problem.
Similar to IRL1... with a key difference:

Lemma

For any $\boldsymbol{x} \in \mathbb{R}^d$, k, γ,

1. *All weights $w^i_{k,\gamma}(\boldsymbol{x}) \in [0, 1]$*
2. $\sum_{i=1}^d w^i_{k,\gamma}(\boldsymbol{x}) = d - k$

Majorization Minimization Scheme

MM scheme to minimize $\mathbf{F}_{\lambda,\gamma}(\boldsymbol{x})$:

$$\mathbf{w}^t = \mathbf{w}_{k,\gamma}(\boldsymbol{x}^{t-1})$$
$$\boldsymbol{x}^t = \arg\min_x \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\langle \mathbf{w}^t, |\boldsymbol{x}|\rangle$$

Each subproblem is a *convex* weighted ℓ_1 problem.
Similar to IRL1... with a key difference:

Lemma

For any $\boldsymbol{x} \in \mathbb{R}^d$, k, γ,

1. *All weights $w^i_{k,\gamma}(\boldsymbol{x}) \in [0,1]$*
2. $\sum_{i=1}^d w^i_{k,\gamma}(\boldsymbol{x}) = d - k$

Since all weights are in [0,1], and their sum is constant, they do not require regularization.

Computing $\tau_{k,\gamma}$ and $\mathbf{w}_{k,\gamma}$

Problem: How to compute $\tau_{k,\gamma}(\boldsymbol{x})$ and $\mathbf{w}_{k,\gamma}(\boldsymbol{x})$?

Their formulas involve sums of $\binom{d}{k}$ terms.

Computing $\tau_{k,\gamma}$ and $\mathbf{w}_{k,\gamma}$

Problem: How to compute $\tau_{k,\gamma}(\boldsymbol{x})$ and $\mathbf{w}_{k,\gamma}(\boldsymbol{x})$?

Their formulas involve sums of $\binom{d}{k}$ terms.

Naïve calculation would be...

Computing $\tau_{k,\gamma}$ and $\mathbf{w}_{k,\gamma}$

Problem: How to compute $\tau_{k,\gamma}(\boldsymbol{x})$ and $\mathbf{w}_{k,\gamma}(\boldsymbol{x})$?

Their formulas involve sums of $\binom{d}{k}$ terms.

Naïve calculation would be...

- *prohibitively* slow.

Computing $\tau_{k,\gamma}$ and $\mathbf{w}_{k,\gamma}$

Problem: How to compute $\tau_{k,\gamma}(\boldsymbol{x})$ and $\mathbf{w}_{k,\gamma}(\boldsymbol{x})$?

Their formulas involve sums of $\binom{d}{k}$ terms.

Naïve calculation would be...

- *prohibitively* slow.
- highly prone to numerical corruption by arithmetic overflow and underflow, due to the log and exp operations.

Computing $\tau_{k,\gamma}$ and $\mathbf{w}_{k,\gamma}$

Problem: How to compute $\tau_{k,\gamma}(\boldsymbol{x})$ and $\mathbf{w}_{k,\gamma}(\boldsymbol{x})$?

Their formulas involve sums of $\binom{d}{k}$ terms.

Naïve calculation would be...

- *prohibitively* slow.
- highly prone to numerical corruption by arithmetic overflow and underflow, due to the log and exp operations.

Developed numerical scheme to accurately compute $\tau_{k,\gamma}(\boldsymbol{x})$ and $\mathbf{w}_{k,\gamma}(\boldsymbol{x})$

Computing $\tau_{k,\gamma}$ and $\mathbf{w}_{k,\gamma}$

Problem: How to compute $\tau_{k,\gamma}(\boldsymbol{x})$ and $\mathbf{w}_{k,\gamma}(\boldsymbol{x})$?

Their formulas involve sums of $\binom{d}{k}$ terms.

Naïve calculation would be...

- *prohibitively* slow.
- highly prone to numerical corruption by arithmetic overflow and underflow, due to the log and exp operations.

Developed numerical scheme to accurately compute $\tau_{k,\gamma}(\boldsymbol{x})$ and $\mathbf{w}_{k,\gamma}(\boldsymbol{x})$

- Recursive, takes $\mathcal{O}(kd)$ operations

Computing $\tau_{k,\gamma}$ and $\mathbf{w}_{k,\gamma}$

Problem: How to compute $\tau_{k,\gamma}(\boldsymbol{x})$ and $\mathbf{w}_{k,\gamma}(\boldsymbol{x})$?

Their formulas involve sums of $\binom{d}{k}$ terms.

Naïve calculation would be...

- *prohibitively* slow.
- highly prone to numerical corruption by arithmetic overflow and underflow, due to the log and exp operations.

Developed numerical scheme to accurately compute $\tau_{k,\gamma}(\boldsymbol{x})$ and $\mathbf{w}_{k,\gamma}(\boldsymbol{x})$

- Recursive, takes $\mathcal{O}(kd)$ operations

Approach also relevant for top-k classification. Method to compute similar functions for small k was proposed by [Berrada, Zisserman, Kumar, *ICLR* '18].

Outline of our method

(a) We seek a solution of (P0) by solving

$$\frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \tau_k(\boldsymbol{x})$$

for increasing values of $\lambda < \bar{\lambda}$, till a k-sparse solution found.

Outline of our method

(a) We seek a solution of (P0) by solving

$$\frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \tau_k(\boldsymbol{x})$$

for increasing values of $\lambda < \bar{\lambda}$, till a k-sparse solution found.

(b) Each such problem solved by homotopy: Minimize

$$\frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \tau_{k,\gamma}(\boldsymbol{x})$$

for increasing sequence of values of γ.

Outline of our method

(a) We seek a solution of (P0) by solving

$$\frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \tau_k(\boldsymbol{x})$$

for increasing values of $\lambda < \bar{\lambda}$, till a k-sparse solution found.

(b) Each such problem solved by homotopy: Minimize

$$\frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \tau_{k,\gamma}(\boldsymbol{x})$$

for increasing sequence of values of γ.

(c) Each such problem solved by MM, requiring solution of several weighted ℓ_1 problems.

Outline of our method

(a) We seek a solution of (P0) by solving

$$\frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_k(\boldsymbol{x})$$

for increasing values of $\lambda < \bar{\lambda}$, till a k-sparse solution found.

(b) Each such problem solved by homotopy: Minimize

$$\frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\tau_{k,\gamma}(\boldsymbol{x})$$

for increasing sequence of values of γ.

(c) Each such problem solved by MM, requiring solution of several weighted ℓ_1 problems.

Running time for one λ: $\approx 500\times$ slower than single ℓ_1 problem.

Comparison to current state of the art

(As in [Bertsimas and Van Parys, 2020])

- $\boldsymbol{x}_0 \in \mathbb{R}^d$ is k-sparse, $d = 15000$, $k = 10$, with entries ± 1
- $A \in \mathbb{R}^{n \times d}$ with uncorrelated $\mathcal{N}(0,1)$ entries

Comparison to current state of the art

(As in [Bertsimas and Van Parys, 2020])

- $\boldsymbol{x}_0 \in \mathbb{R}^d$ is k-sparse, $d = 15000$, $k = 10$, with entries ± 1
- $A \in \mathbb{R}^{n \times d}$ with uncorrelated $\mathcal{N}(0, 1)$ entries
- Observation: $\mathbf{y} = A\boldsymbol{x}_0 + \mathbf{e}$, with 5% noise (SNR=400)
- True k is known to all methods

Comparison to current state of the art

(As in [Bertsimas and Van Parys, 2020])

- $\boldsymbol{x}_0 \in \mathbb{R}^d$ is k-sparse, $d = 15000$, $k = 10$, with entries ± 1
- $A \in \mathbb{R}^{n \times d}$ with uncorrelated $\mathcal{N}(0, 1)$ entries
- Observation: $\mathbf{y} = A\boldsymbol{x}_0 + \mathbf{e}$, with 5% noise (SNR=400)
- True k is known to all methods
- Coordinate descent returns multiple solutions

  Chose the one whose support is closest to the true support

Comparison to current state of the art

(As in [Bertsimas and Van Parys, 2020])

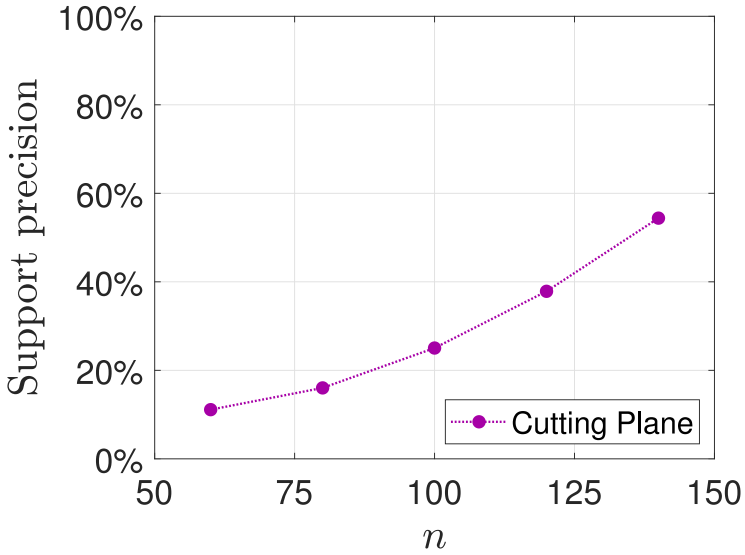
- $\boldsymbol{x}_0 \in \mathbb{R}^d$ is k-sparse, $d = 15000$, $k = 10$, with entries ± 1
- $A \in \mathbb{R}^{n\times d}$ with uncorrelated $\mathcal{N}(0,1)$ entries
- Observation: $\mathbf{y} = A\boldsymbol{x}_0 + \mathbf{e}$, with 5% noise (SNR=400)
- True k is known to all methods
- Coordinate descent returns multiple solutions

  Chose the one whose support is closest to the true support

Measure of success:

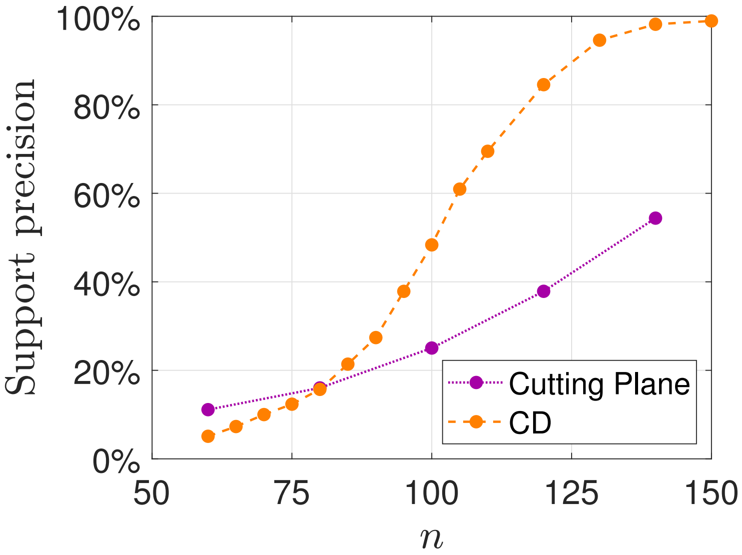
- Support accuracy: $\frac{|\hat{S} \cap S_0|}{k}$

Comparison to current state of the art



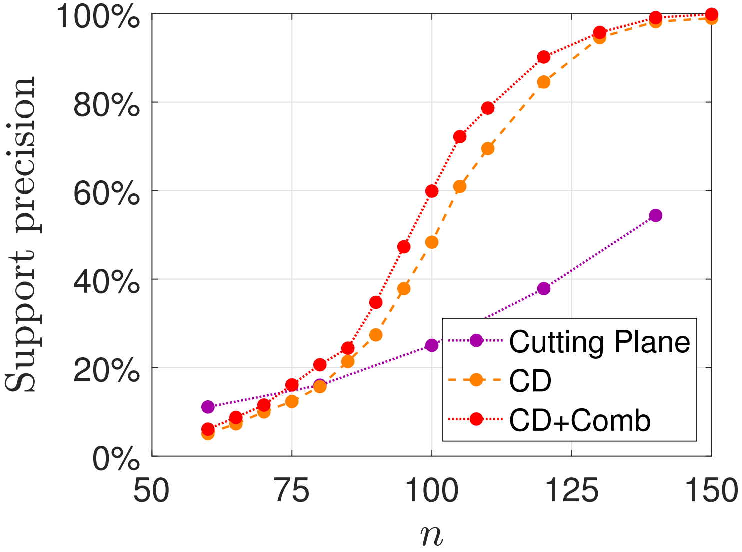
$k = 10$, $d = 15000$, 5% noise (SNR=400)

Comparison to current state of the art



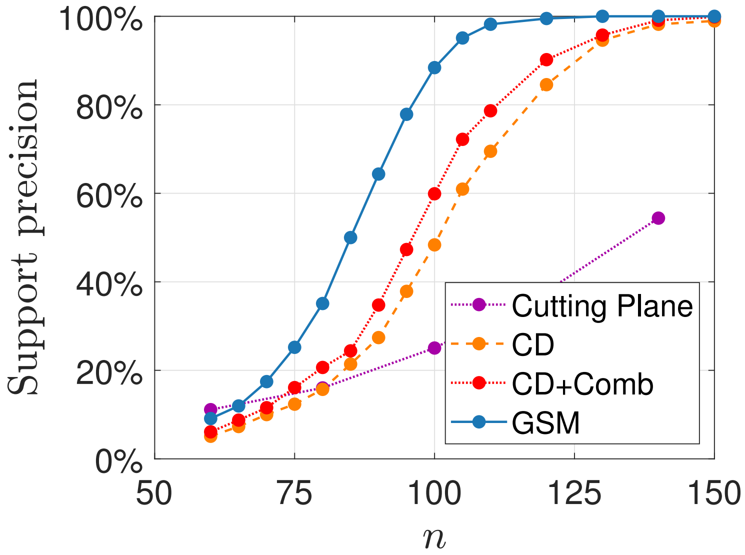
$k = 10$, $d = 15000$, 5% noise (SNR=400)

Comparison to current state of the art



$k = 10$, $d = 15000$, 5% noise (SNR=400)

Comparison to current state of the art



$k = 10$, $d = 15000$, 5% noise (SNR=400)

Comparison to current state of the art

(As in [Hazimeh, Mazumder 2020])

- k-sparse signal $\boldsymbol{x}_0 \in \mathbb{R}^d$, $k = 50$, $d = 20000$
- Entries ± 1

Comparison to current state of the art

(As in [Hazimeh, Mazumder 2020])

- k-sparse signal $\boldsymbol{x}_0 \in \mathbb{R}^d$, $k = 50$, $d = 20000$
- Entries ± 1
- $A \in \mathbb{R}^{n \times d}$, $\mathcal{N}(0, \Sigma)$, $\Sigma_{i,j} = 0.5^{|i-j|}$
- Observation: $\mathbf{y} = A\boldsymbol{x}_0 + \mathbf{e}$, varying noise levels

Comparison to current state of the art

(As in [Hazimeh, Mazumder 2020])

- k-sparse signal $\boldsymbol{x}_0 \in \mathbb{R}^d$, $k = 50$, $d = 20000$
- Entries ± 1
- $A \in \mathbb{R}^{n \times d}$, $\mathcal{N}(0, \Sigma)$, $\Sigma_{i,j} = 0.5^{|i-j|}$
- Observation: $\mathbf{y} = A\boldsymbol{x}_0 + \mathbf{e}$, varying noise levels
- Each method chooses k using a separate validation set: $\tilde{\mathbf{y}} = \tilde{A}\boldsymbol{x}_0 + \tilde{\mathbf{e}}$

Comparison to current state of the art

(As in [Hazimeh, Mazumder 2020])

- k-sparse signal $\boldsymbol{x}_0 \in \mathbb{R}^d$, $k = 50$, $d = 20000$
- Entries ± 1
- $A \in \mathbb{R}^{n \times d}$, $\mathcal{N}(0, \Sigma)$, $\Sigma_{i,j} = 0.5^{|i-j|}$
- Observation: $\mathbf{y} = A\boldsymbol{x}_0 + \mathbf{e}$, varying noise levels
- Each method chooses k using a separate validation set: $\tilde{\mathbf{y}} = \tilde{A}\boldsymbol{x}_0 + \tilde{\mathbf{e}}$

Measures of success:

Comparison to current state of the art

(As in [Hazimeh, Mazumder 2020])

- k-sparse signal $\boldsymbol{x}_0 \in \mathbb{R}^d$, $k = 50$, $d = 20000$
- Entries ± 1
- $A \in \mathbb{R}^{n \times d}$, $\mathcal{N}(0, \Sigma)$, $\Sigma_{i,j} = 0.5^{|i-j|}$
- Observation: $\mathbf{y} = A\boldsymbol{x}_0 + \mathbf{e}$, varying noise levels
- Each method chooses k using a separate validation set: $\tilde{\mathbf{y}} = \tilde{A}\boldsymbol{x}_0 + \tilde{\mathbf{e}}$

Measures of success:

- F-score: $2\frac{|\hat{S} \cap S_0|}{|\hat{S}| + |S_0|}$

Comparison to current state of the art

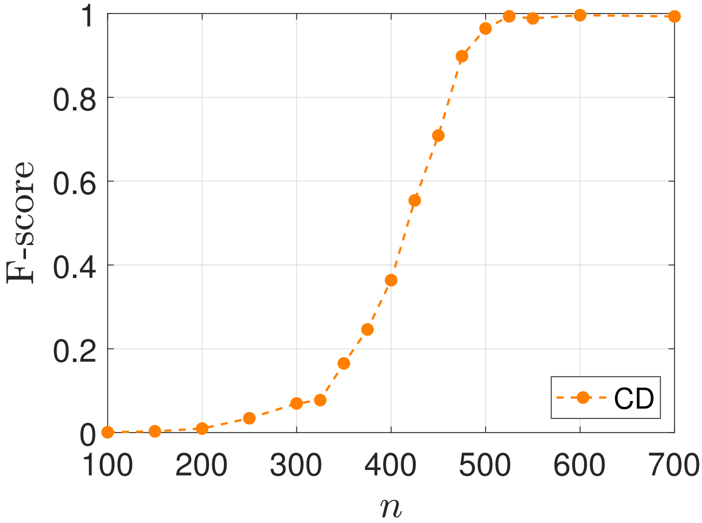
(As in [Hazimeh, Mazumder 2020])

- k-sparse signal $\boldsymbol{x}_0 \in \mathbb{R}^d$, $k = 50$, $d = 20000$
- Entries ± 1
- $A \in \mathbb{R}^{n \times d}$, $\mathcal{N}(0, \Sigma)$, $\Sigma_{i,j} = 0.5^{|i-j|}$
- Observation: $\mathbf{y} = A\boldsymbol{x}_0 + \mathbf{e}$, varying noise levels
- Each method chooses k using a separate validation set: $\tilde{\mathbf{y}} = \tilde{A}\boldsymbol{x}_0 + \tilde{\mathbf{e}}$

Measures of success:

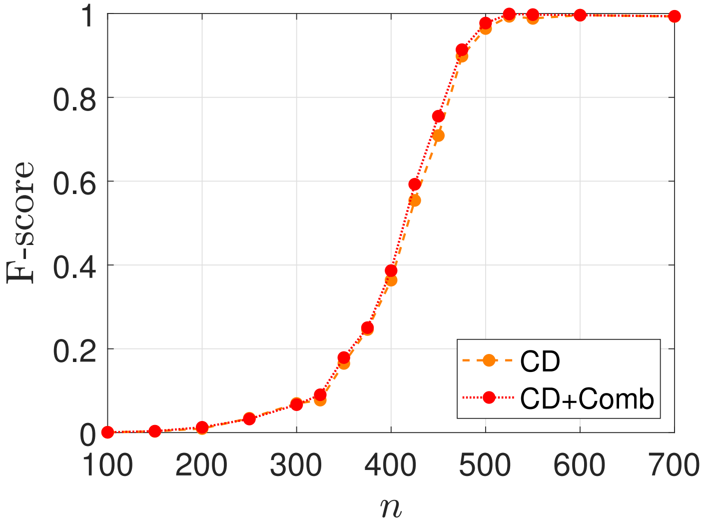
- F-score: $2\frac{|\hat{S} \cap S_0|}{|\hat{S}| + |S_0|}$
- Expected prediction error: $\sqrt{\frac{\mathbb{E}_{\mathbf{A},\mathbf{y}}\left[\left\|A\hat{\boldsymbol{x}} - y\right\|^2\right]}{\mathbb{E}_{\mathbf{y}}\left[\|y\|^2\right]}}$

Comparison to current state of the art



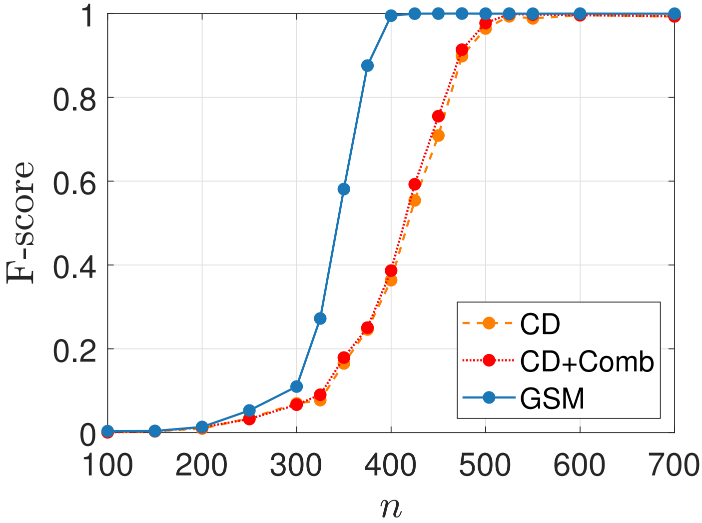
$d = 20000$ $k = 50$ Entries: ± 1 5% noise (SNR=400)

Comparison to current state of the art



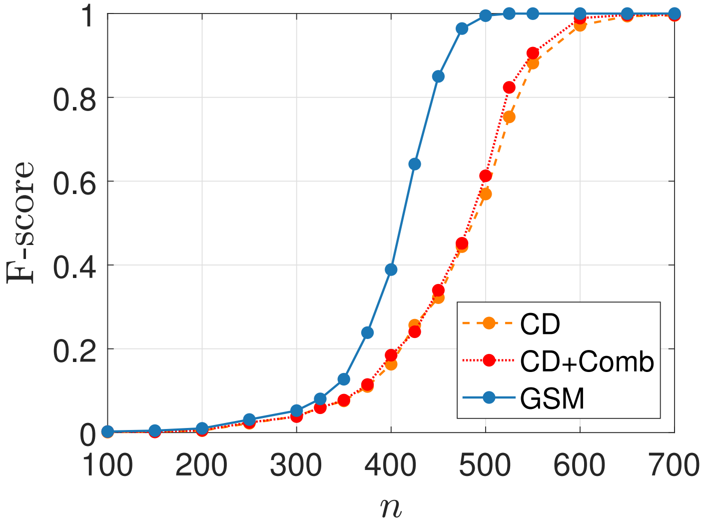
$d = 20000 \quad k = 50$ Entries: ± 1 5% noise (SNR=400)

Comparison to current state of the art



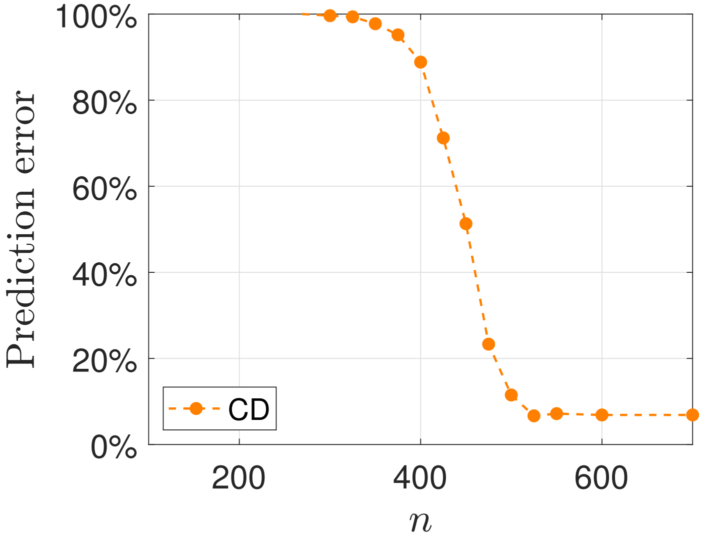
$d = 20000$ $k = 50$ Entries: ± 1 5% noise (SNR=400)

Comparison to current state of the art



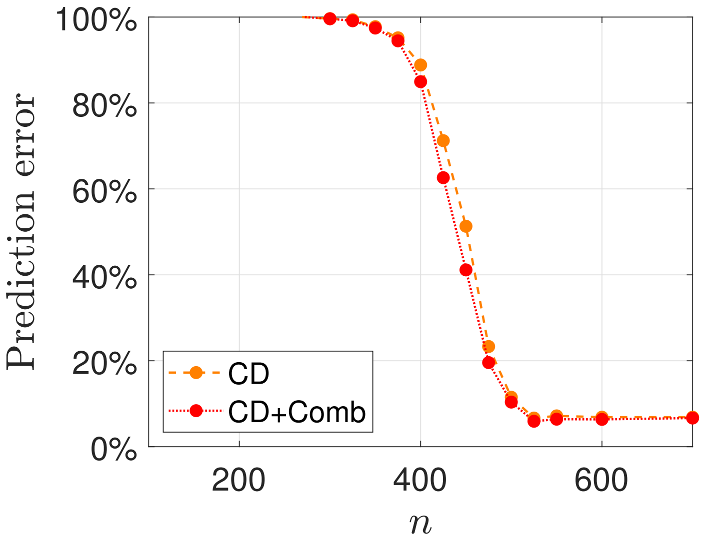
$d = 20000$ $k = 50$ Entries: ± 1 33.3% noise (SNR=9)

Comparison to current state of the art



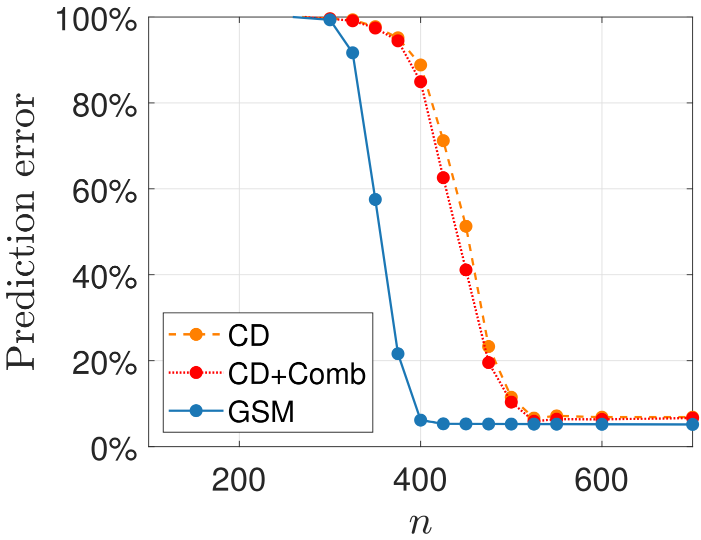
$d = 20000$ $k = 50$ Entries: ± 1 5% noise (SNR=400)

Comparison to current state of the art



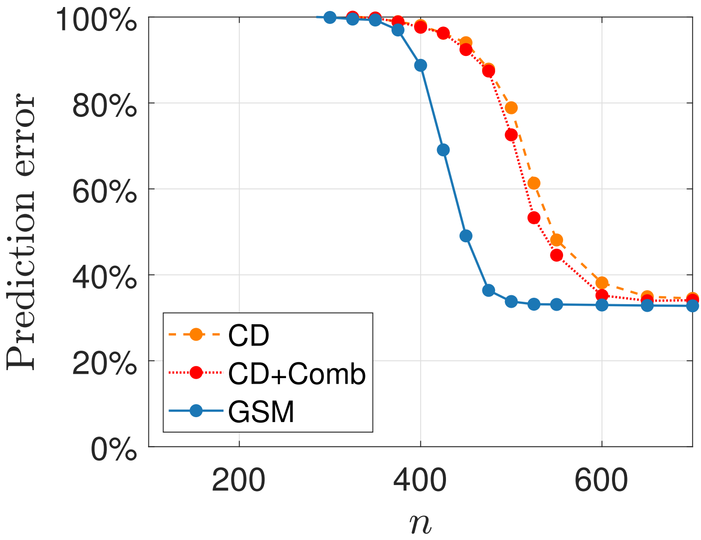
$d = 20000$ $k = 50$ Entries: ± 1 5% noise (SNR=400)

Comparison to current state of the art



$d = 20000$ $k = 50$ Entries: ± 1 5% noise (SNR=400)

Comparison to current state of the art



$d = 20000$ $k = 50$ Entries: ± 1 33.3% noise (SNR=9)

Conclusion

- Problem (P0) plays a key role in multiple applications.
- Still room for improvements for challenging instances of (P0)

Conclusion

- Problem (P0) plays a key role in multiple applications.
- Still room for improvements for challenging instances of (P0)
- Trimmed Lasso - desirable theoretical properties to solve (P0)

Conclusion

- Problem (P0) plays a key role in multiple applications.
- Still room for improvements for challenging instances of (P0)
- Trimmed Lasso - desirable theoretical properties to solve (P0)
- Practical optimization method for Trimmed-Lasso penalty

Conclusion

- Problem (P0) plays a key role in multiple applications.
- Still room for improvements for challenging instances of (P0)
- Trimmed Lasso - desirable theoretical properties to solve (P0)
- Practical optimization method for Trimmed-Lasso penalty
  - Novel surrogate penalty (GSM)
  - Accurate numerical scheme
  - Accompanying optimization algorithm
- Approach potentially applicable to other sparse combinatorial search problems

Conclusion

- Problem (P0) plays a key role in multiple applications.
- Still room for improvements for challenging instances of (P0)
- Trimmed Lasso - desirable theoretical properties to solve (P0)
- Practical optimization method for Trimmed-Lasso penalty
  - Novel surrogate penalty (GSM)
  - Accurate numerical scheme
  - Accompanying optimization algorithm
- Approach potentially applicable to other sparse combinatorial search problems

code on GitHub.

Amir, T., Basri, R. and Nadler, B., The Trimmed Lasso: Sparse Recovery Guarantees and Practical Optimization by the Generalized Soft-Min Penalty. *SIAM J. Math. Data Science, 2021*

Thank You

The End