# Sparse Algorithms are not Stable: a no free lunch theorem

Huan Xu
National University of Singapore,
Dept. of Mechanical Engineering & Dept. of Mathematics (by courtesy)

Joint work with: Constantine Caramanis† and Shie Mannor‡
† UT-Austin / ‡ Technion

Dec 8, 2012

## The underlying problem

- The Regression problem:
  - Given iid points $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N$;
  - Given labels $b_i$ for each point $\boldsymbol{a}_i$;
  - Goal: determine a regressor (a linear function) to learn the (linear) relationship between data points and points generated.

- Thus we have: Learning algorithm $\mathbb{L}$

$$\mathbb{L} : (\boldsymbol{b}, A) \longmapsto \boldsymbol{x}^*,$$

- Loss: $|\boldsymbol{a}^\top \boldsymbol{x}^* - b|$. Notation: $|\mathbb{L}_{(\boldsymbol{b},A)}(\boldsymbol{a}) - b|$.
- Applications.

## Sparsity and Stability

- Properties of the solution: We want an algorithm $\mathbb{L}$ that.

## Sparsity and Stability

- Properties of the solution: We want an algorithm $\mathbb{L}$ that.
  - Minimize some notion of loss, e.g., $||A\boldsymbol{x} - \boldsymbol{b}||_2$, or $\mathbb{E}(|\mathbb{L}_{(\boldsymbol{b},A)}(\boldsymbol{a}) - b|)$.

## Sparsity and Stability

- Properties of the solution: We want an algorithm $\mathbb{L}$ that.
  - Minimize some notion of loss, e.g., $||A\boldsymbol{x} - \boldsymbol{b}||_2$, or $\mathbb{E}(|\mathbb{L}_{(\boldsymbol{b},A)}(\boldsymbol{a}) - b|)$.
  - Other properties?

## Sparsity and Stability

- Properties of the solution: We want an algorithm $\mathbb{L}$ that.
  - Minimize some notion of loss, e.g., $||A\boldsymbol{x} - \boldsymbol{b}||_2$, or $\mathbb{E}(|\mathbb{L}_{(\boldsymbol{b},A)}(\boldsymbol{a}) - b|)$.
  - Other properties?
  - Sparsity:
    - Recover an accurate solution with few nonzero entries.
    - Identify the sparsity pattern of generative model.
    - Why: dimensionality reduction; interpretability; etc.

## Sparsity and Stability

- Properties of the solution: We want an algorithm $\mathbb{L}$ that.
  - Minimize some notion of loss, e.g., $||A\boldsymbol{x} - \boldsymbol{b}||_2$, or $\mathbb{E}(|\mathbb{L}_{(\boldsymbol{b},A)}(\boldsymbol{a}) - b|)$.
  - Other properties?
  - Sparsity:
    - Recover an accurate solution with few nonzero entries.
    - Identify the sparsity pattern of generative model.
    - Why: dimensionality reduction; interpretability; etc.
  - Stability:
    - $\mathbb{L}$: this is a function of the data, $(\boldsymbol{b}, A)$.
    - What if data change a little?
    - Tool to prove performance bounds.

## Sparsity of Solution

- Some reasons sparsity is desirable
  - Underdetermined system of equations (signal processing).
  - Model identification.
  - Algorithmic: Efficient evaluation.
- Some algorithms that *encourage* sparsity
  - Regression + explicit sparsity constraint (NP-hard).
  - Lasso: $\ell^1$-regularized regression: $\min : ||A\boldsymbol{x} - \boldsymbol{b}||_2^2 + \lambda||\boldsymbol{x}||_1$
  - Huge literature: Tibshirani, Candes, Tao, Donoho, + many, many others.

## Stability of Solution

- Definition of Algorithmic Stability:
  - Intuitive definition: Stable if $\mathbb{L}_{(\mathbf{b},A)}$ and $\mathbb{L}_{(\mathbf{b},A)^{\setminus i}}$ are close.

## Stability of Solution

- Definition of Algorithmic Stability:
  - Intuitive definition: Stable if $\mathbb{L}_{(\mathbf{b},A)}$ and $\mathbb{L}_{(\mathbf{b},A)^{\setminus i}}$ are close.
  - Formal definition: An algorithm $\mathbb{L}$ has uniform stability $\beta_n$ with respect to the loss function $l$ if the following holds:

$$\forall(\mathbf{b},A) \in \mathcal{Z}^n, \forall i \in \{1,\cdots,n\}, \forall \mathbf{z} \in \mathcal{Z}$$
$$|l(\mathbb{L}_{(\mathbf{b},A)}, \mathbf{z}) - l(\mathbb{L}_{(\mathbf{b},A)^{\setminus i}}, \mathbf{z})| \leq \beta_n.$$

[Devroye & Wagner 1979, Bousquet & Elisseeff 2002]

## Stability of Solution

- Definition of Algorithmic Stability:
  - Intuitive definition: Stable if $\mathbb{L}_{(\mathbf{b},A)}$ and $\mathbb{L}_{(\mathbf{b},A)^{\setminus i}}$ are close.
  - Formal definition: An algorithm $\mathbb{L}$ has uniform stability $\beta_n$ with respect to the loss function $l$ if the following holds:

$$\forall(\mathbf{b},A) \in \mathcal{Z}^n, \forall i \in \{1,\cdots,n\}, \forall \mathbf{z} \in \mathcal{Z}$$
$$|l(\mathbb{L}_{(\mathbf{b},A)}, \mathbf{z}) - l(\mathbb{L}_{(\mathbf{b},A)^{\setminus i}}, \mathbf{z})| \leq \beta_n.$$

    [Devroye & Wagner 1979, Bousquet & Elisseeff 2002]

- Example: $\ell^2$-regularized regression:
  - Algorithm given by: $\min : ||A\boldsymbol{x} - \boldsymbol{b}||_2^2 + \lambda||\boldsymbol{x}||_2$
  - Stability: $\beta_n = O(1/n)$.

## Stability of Solution

- Definition of Algorithmic Stability:
  - Intuitive definition: Stable if $\mathbb{L}_{(\mathbf{b},A)}$ and $\mathbb{L}_{(\mathbf{b},A)^{\setminus i}}$ are close.
  - Formal definition: An algorithm $\mathbb{L}$ has uniform stability $\beta_n$ with respect to the loss function $l$ if the following holds:

$$\forall(\mathbf{b},A) \in \mathcal{Z}^n, \forall i \in \{1,\cdots,n\}, \forall \mathbf{z} \in \mathcal{Z}$$
$$|l(\mathbb{L}_{(\mathbf{b},A)}, \mathbf{z}) - l(\mathbb{L}_{(\mathbf{b},A)^{\setminus i}}, \mathbf{z})| \leq \beta_n.$$

    [Devroye & Wagner 1979, Bousquet & Elisseeff 2002]

- Example: $\ell^2$-regularized regression:
  - Algorithm given by: $\min : ||A\boldsymbol{x} - \boldsymbol{b}||_2^2 + \lambda||\boldsymbol{x}||_2$
  - Stability: $\beta_n = O(1/n)$.

- PAC bounds: $n$ samples, $\beta$ stability, $M$ bound on loss:

$$R \leq R_{\text{emp}} + 2\beta + (4n\beta + M)\sqrt{\frac{\ln 1/\delta}{2n}}.$$

## This Talk: Can't Have Both

- Sparse algorithms are not stable:
  - Given $(\boldsymbol{b}, A)$, an algorithm is sparse if it can identify redundant features: when features $\boldsymbol{A}_i = \boldsymbol{A}_j$, either $x_i$ or $x_j$ is zero. (weak notion).
  - If an algorithm has this property, then $\beta_n = \Omega(1)$ – it does not decrease with the number of samples.
- Stable algorithms are not sparse.
  - An algorithm is stable if $\beta_n \to 0$ as $n \to \infty$.
  - If an algorithm has stability with $\beta_n \to 0$, then it does not have the above sparsity property

## This Talk: Can't Have Both

- Sparse algorithms are not stable:
  - Given $(\boldsymbol{b}, A)$, an algorithm is sparse if it can identify redundant features: when features $\boldsymbol{A}_i = \boldsymbol{A}_j$, either $x_i$ or $x_j$ is zero. (weak notion).
  - If an algorithm has this property, then $\beta_n = \Omega(1)$ – it does not decrease with the number of samples.
- Stable algorithms are not sparse.
  - An algorithm is stable if $\beta_n \to 0$ as $n \to \infty$.
  - If an algorithm has stability with $\beta_n \to 0$, then it does not have the above sparsity property
- Plan from here:
  - Prove for the special case of the Lasso algorithm.
  - Prove for the general case.

## Lasso is not stable

- Lasso algorithm: Given $(\boldsymbol{b}, A)$, selected regressor is:

$$\boldsymbol{x}^* = \operatorname{argmin} ||A\boldsymbol{x} - \boldsymbol{b}||_2^2 + \lambda||\boldsymbol{x}||_1 .$$

## Lasso is not stable

- Lasso algorithm: Given $(\boldsymbol{b}, A)$, selected regressor is:

$$\boldsymbol{x}^* = \operatorname{argmin} ||A\boldsymbol{x} - \boldsymbol{b}||_2^2 + \lambda||\boldsymbol{x}||_1 .$$

- **Theorem**. For $\mathbb{L}$ the Lasso algorithm, its stability parameter is lower bounded by a constant.
  - That constant is independent of the number of samples.
  - That constant is lower bounded by the worst-case loss of Lasso, when trained on the worst possible set, and evaluated on the worst possible point.

## Lasso is not stable

- Lasso algorithm: Given $(\boldsymbol{b}, A)$, selected regressor is:

$$\boldsymbol{x}^* = \operatorname{argmin} ||A\boldsymbol{x} - \boldsymbol{b}||_2^2 + \lambda ||\boldsymbol{x}||_1 .$$

- **Theorem**. For $\mathbb{L}$ the Lasso algorithm, its stability parameter is lower bounded by a constant.
    - That constant is independent of the number of samples.
    - That constant is lower bounded by the worst-case loss of Lasso, when trained on the worst possible set, and evaluated on the worst possible point.
- The “pseudo maximal error”:

## Lasso is not stable

- Lasso algorithm: Given $(\boldsymbol{b}, A)$, selected regressor is:

$$\boldsymbol{x}^* = \operatorname{argmin} ||A\boldsymbol{x} - \boldsymbol{b}||_2^2 + \lambda||\boldsymbol{x}||_1.$$

- **Theorem**. For $\mathbb{L}$ the Lasso algorithm, its stability parameter is lower bounded by a constant.
  - That constant is independent of the number of samples.
  - That constant is lower bounded by the worst-case loss of Lasso, when trained on the worst possible set, and evaluated on the worst possible point.

- The "pseudo maximal error": Given the sample space $\mathcal{Z} = \mathcal{Y} \times \mathcal{X}$ where $\mathcal{Y} \subseteq \mathbb{R}$, $\mathcal{X} \subseteq \mathbb{R}^m$, and $0 \in \mathcal{Y}$. The pseudo maximal error for a learning algorithm $\mathbb{L}$ w.r.t. $\mathcal{Z}$ is

$$\mathfrak{b}_n(\mathbb{L}, \mathcal{Z}) \triangleq \max_{(\mathbf{b},A)\in\mathcal{Z}^n, \mathbf{z}\in\mathcal{X}} l\Big(\mathbb{L}_{(\mathbf{b},A)}, (0, \mathbf{z})\Big).$$

- PME does not decrease with $n$.

## Lasso is not stable

- $\mathcal{Z} = \mathcal{Y} \times \mathcal{X}$ is the sample space with $m$ features.
- $\hat{\mathcal{Z}} \subseteq \mathcal{Y} \times \mathcal{X} \times \mathcal{X}$ is the sample space with with $2m$ features.

**Theorem.** The uniform stability bound $\beta$ of Lasso trained on $n$ points of $\hat{\mathcal{Z}}$, is lower bounded by $\mathfrak{b}_n(\text{Lasso}, \mathcal{Z})$.

## Lasso is not stable: proof

- Choose $(\mathbf{b}^*, A^*)$ and $(0, \mathbf{z}^{*\top})$ to jointly achieve lower bound:

$$\mathfrak{b}_n(\mathrm{Lasso}, \mathcal{Z}) = |\mathbb{L}_{(\mathbf{b}, A)}(\boldsymbol{z})|.$$

## Lasso is not stable: proof

- Choose $(\mathbf{b}^*, A^*)$ and $(0, \mathbf{z}^{*\top})$ to jointly achieve lower bound:

$$\mathfrak{b}_n(\mathrm{Lasso}, \mathcal{Z}) = |\mathbb{L}_{(\mathbf{b}, A)}(\boldsymbol{z})|.$$

- Let $\mathbf{x}^*$ be the optimal solution for $(\mathbf{b}^*, A^*)$.

## Lasso is not stable: proof

- Choose $(\mathbf{b}^*, A^*)$ and $(0, \mathbf{z}^{*\top})$ to jointly achieve lower bound:

$$\mathfrak{b}_n(\mathrm{Lasso}, \mathcal{Z}) = |\mathbb{L}_{(\mathbf{b},A)}(\boldsymbol{z})|.$$

- Let $\mathbf{x}^*$ be the optimal solution for $(\mathbf{b}^*, A^*)$.
- Consider the sample set

$$\left( \begin{pmatrix} \mathbf{b}^* \\ 0 \end{pmatrix} \quad \begin{pmatrix} A^* & A^* \\ \mathbf{0}^\top & \mathbf{z}^{*\top} \end{pmatrix} \right)$$

## Lasso is not stable: proof

- Choose $(\mathbf{b}^*, A^*)$ and $(0, \mathbf{z}^{*\top})$ to jointly achieve lower bound:

$$\mathfrak{b}_n(\mathrm{Lasso}, \mathcal{Z}) = |\mathbb{L}_{(\mathbf{b},A)}(\boldsymbol{z})|.$$

- Let $\mathbf{x}^*$ be the optimal solution for $(\mathbf{b}^*, A^*)$.
- Consider the sample set

$$\left( \begin{pmatrix} \mathbf{b}^* \\ 0 \end{pmatrix} \quad \begin{pmatrix} A^* & A^* \\ \mathbf{0}^\top & \mathbf{z}^{*\top} \end{pmatrix} \right)$$

- $(\mathbf{x}, \mathbf{0})$ is an optimal solution of Lasso w.r.t to this sample set.

## Lasso is not stable: proof

- Choose $(\mathbf{b}^*, A^*)$ and $(0, \mathbf{z}^{*\top})$ to jointly achieve lower bound:

$$\mathfrak{b}_n(\mathrm{Lasso}, \mathcal{Z}) = |\mathbb{L}_{(\mathbf{b},A)}(\boldsymbol{z})|.$$

- Let $\mathbf{x}^*$ be the optimal solution for $(\mathbf{b}^*, A^*)$.
- Consider the sample set

$$\left( \begin{pmatrix} \mathbf{b}^* \\ 0 \end{pmatrix} \quad \begin{pmatrix} A^* & A^* \\ \mathbf{0}^\top & \mathbf{z}^{*\top} \end{pmatrix} \right)$$

- $(\mathbf{x}, \mathbf{0})$ is an optimal solution of Lasso w.r.t to this sample set.
- Remove the last sample. Now $(\mathbf{0}, \mathbf{x})$ is an optimal solution.

## Lasso is not stable: proof

- Choose $(\mathbf{b}^*, A^*)$ and $(0, \mathbf{z}^{*\top})$ to jointly achieve lower bound:

$$\mathfrak{b}_n(\mathrm{Lasso}, \mathcal{Z}) = |\mathbb{L}_{(\mathbf{b},A)}(\boldsymbol{z})|.$$

- Let $\mathbf{x}^*$ be the optimal solution for $(\mathbf{b}^*, A^*)$.
- Consider the sample set

$$\left( \begin{pmatrix} \mathbf{b}^* \\ 0 \end{pmatrix} \quad \begin{pmatrix} A^* & A^* \\ \mathbf{0}^\top & \mathbf{z}^{*\top} \end{pmatrix} \right)$$

- $(\mathbf{x}, \mathbf{0})$ is an optimal solution of Lasso w.r.t to this sample set.
- Remove the last sample. Now $(\mathbf{0}, \mathbf{x})$ is an optimal solution.
- Use last sample as a testing observation. Solution w.r.t the full sample set has zero cost, while the solution of the leave-one-out sample set has a cost $\mathfrak{b}_n(\mathrm{Lasso}, \mathcal{Z})$.

## A more general statement/proof

- Is this an artifact of the Lasso algorithm?

## A more general statement/proof

- Is this an artifact of the Lasso algorithm?
- We can generalize:

## A more general statement/proof

- Is this an artifact of the Lasso algorithm?
- We can generalize:
  - Notion of loss function: $l(\mathbb{L}_{(\boldsymbol{b},A)}, (b, \boldsymbol{z}))$.

## A more general statement/proof

- Is this an artifact of the Lasso algorithm?
- We can generalize:
  - Notion of loss function: $l(\mathbb{L}_{(\boldsymbol{b},A)}, (b, \boldsymbol{z}))$.
  - Notion of Sparsity:
    - **Identifying Redundant Features**: A weight vector $\mathbf{w}^*$ *Identifies Redundant Features of* $A$ if

      $$\forall i \neq j, \quad \mathbf{a}_i = \mathbf{a}_j \Rightarrow w_i^* w_j^* = 0.$$

      An algorithm $\mathbb{L}$ is said to be *able to Identify Redundant Features* if $\forall(\mathbf{b}, A)$ there exists $\mathbf{w}^* \in \mathbb{L}_{(\mathbf{b},A)}$ that identifies redundant features of $A$.

## A more general statement/proof

- Is this an artifact of the Lasso algorithm?
- We can generalize:
  - Notion of loss function: $l(\mathbb{L}_{(\boldsymbol{b},A)}, (b, \boldsymbol{z}))$.
  - Notion of Sparsity:
    - **Identifying Redundant Features**: A weight vector $\mathbf{w}^*$ *Identifies Redundant Features of* $A$ if
      $$\forall i \neq j, \quad \mathbf{a}_i = \mathbf{a}_j \Rightarrow w_i^* w_j^* = 0.$$
      An algorithm $\mathbb{L}$ is said to be *able to Identify Redundant Features* if $\forall(\mathbf{b}, A)$ there exists $\mathbf{w}^* \in \mathbb{L}_{(\mathbf{b},A)}$ that identifies redundant features of $A$.
  - Notion of algorithm.
    - Learning algorithm: $\mathbb{L}_{(\boldsymbol{b},A)}$.
    - Output of learning algorithm: regressor
    - Need to talk generically about an optimal solution.

## Definitions for the proof

- Given input data, $(\boldsymbol{b}, A)$, a learning algorithm $\mathbb{L}$ induces partial ordering on solutions:

$$\boldsymbol{x}^1 \preceq_{(\mathbf{b},A)} \boldsymbol{x}^2 : \quad \text{Algorithm } \mathbb{L} \text{ prefers } \boldsymbol{x}^2 \text{ to } \boldsymbol{x}^1.$$

- Some assumptions:

## Definitions for the proof

- Given input data, $(\boldsymbol{b}, A)$, a learning algorithm $\mathbb{L}$ induces partial ordering on solutions:

$$\boldsymbol{x}^1 \preceq_{(\mathbf{b},A)} \boldsymbol{x}^2 : \quad \text{Algorithm } \mathbb{L} \text{ prefers } \boldsymbol{x}^2 \text{ to } \boldsymbol{x}^1.$$

- Some assumptions:

  (i) Given $j$, $\mathbf{b}$, $A$, $\mathbf{x}^1$ and $\mathbf{x}^2$, if $\mathbf{x}^1 \preceq_{(\mathbf{b},A)} \mathbf{x}^2$, and $x_j^1 = x_j^2 = 0$, then for any $\hat{\mathbf{a}}$

$$\mathbf{x}^1 \preceq_{(\mathbf{b},\hat{A})} \mathbf{x}^2, \quad \text{where } \hat{A} = (\mathbf{a}_1, \cdots, \mathbf{a}_{j-1}, \hat{\mathbf{a}}, \mathbf{a}_{j+1}, \cdots, \mathbf{a}_m).$$

## Definitions for the proof

- Given input data, $(\boldsymbol{b}, A)$, a learning algorithm $\mathbb{L}$ induces partial ordering on solutions:

$$\boldsymbol{x}^1 \preceq_{(\mathbf{b},A)} \boldsymbol{x}^2 : \quad \text{Algorithm } \mathbb{L} \text{ prefers } \boldsymbol{x}^2 \text{ to } \boldsymbol{x}^1.$$

- Some assumptions:

  (i) Given $j$, $\mathbf{b}$, $A$, $\mathbf{x}^1$ and $\mathbf{x}^2$, if $\mathbf{x}^1 \preceq_{(\mathbf{b},A)} \mathbf{x}^2$, and $x_j^1 = x_j^2 = 0$, then for any $\hat{\mathbf{a}}$

$$\mathbf{x}^1 \preceq_{(\mathbf{b},\hat{A})} \mathbf{x}^2, \quad \text{where } \hat{A} = (\mathbf{a}_1, \cdots, \mathbf{a}_{j-1}, \hat{\mathbf{a}}, \mathbf{a}_{j+1}, \cdots, \mathbf{a}_m)\,.$$

  (ii) Given $\mathbf{b}$, $A$, $\mathbf{x}^1$, $\mathbf{x}^2$, $b'$ and $\mathbf{z}$, if $\mathbf{x}^1 \preceq_{(\mathbf{b},A)} \mathbf{x}^2$, $\quad b = \mathbf{z}^\top \mathbf{x}^2$ then

$$\mathbf{x}^1 \preceq_{(\overline{\mathbf{b}},\overline{A})} \mathbf{x}^2, \quad \text{where } \overline{\mathbf{b}} = \begin{pmatrix} \mathbf{b} \\ b' \end{pmatrix}; \quad \overline{A} = \begin{pmatrix} A \\ \mathbf{z}^\top \end{pmatrix}.$$

## Definitions for the proof

- Given input data, $(\boldsymbol{b}, A)$, a learning algorithm $\mathbb{L}$ induces partial ordering on solutions:

$$\boldsymbol{x}^1 \preceq_{(\mathbf{b},A)} \boldsymbol{x}^2 : \quad \text{Algorithm } \mathbb{L} \text{ prefers } \boldsymbol{x}^2 \text{ to } \boldsymbol{x}^1.$$

- Some assumptions:

(i) Given $j$, $\mathbf{b}$, $A$, $\mathbf{x}^1$ and $\mathbf{x}^2$, if $\mathbf{x}^1 \preceq_{(\mathbf{b},A)} \mathbf{x}^2$, and $x_j^1 = x_j^2 = 0$, then for any $\hat{\mathbf{a}}$

$$\mathbf{x}^1 \preceq_{(\mathbf{b},\hat{A})} \mathbf{x}^2, \quad \text{where } \hat{A} = (\mathbf{a}_1, \cdots, \mathbf{a}_{j-1}, \hat{\mathbf{a}}, \mathbf{a}_{j+1}, \cdots, \mathbf{a}_m)\,.$$

(ii) Given $\mathbf{b}$, $A$, $\mathbf{x}^1$, $\mathbf{x}^2$, $b'$ and $\mathbf{z}$, if $\mathbf{x}^1 \preceq_{(\mathbf{b},A)} \mathbf{x}^2$, $\quad b = \mathbf{z}^\top \mathbf{x}^2$ then

$$\mathbf{x}^1 \preceq_{(\overline{\mathbf{b}},\overline{A})} \mathbf{x}^2, \quad \text{where } \overline{\mathbf{b}} = \begin{pmatrix} \mathbf{b} \\ b' \end{pmatrix}; \quad \overline{A} = \begin{pmatrix} A \\ \mathbf{z}^\top \end{pmatrix}.$$

(iii) Given $j$, $\mathbf{b}$, $A$, $\mathbf{x}^1$ and $\mathbf{x}^2$, if $\mathbf{x}^1 \preceq_{(\mathbf{b},A)} \mathbf{x}^2$, then

$$\hat{\mathbf{x}}^1 \preceq_{(\mathbf{b},\tilde{A})} \hat{\mathbf{x}}^2, \quad \text{where } \hat{\mathbf{x}}^i = \begin{pmatrix} \mathbf{x}^i \\ 0 \end{pmatrix},\ i = 1, 2; \quad \tilde{A} = (A, \mathbf{0})\,.$$

## Definitions for the proof

- Given input data, $(\boldsymbol{b}, A)$, a learning algorithm $\mathbb{L}$ induces partial ordering on solutions:

$$\boldsymbol{x}^1 \preceq_{(\mathbf{b},A)} \boldsymbol{x}^2 : \quad \text{Algorithm } \mathbb{L} \text{ prefers } \boldsymbol{x}^2 \text{ to } \boldsymbol{x}^1.$$

- Some assumptions:

(i) Given $j$, $\mathbf{b}$, $A$, $\mathbf{x}^1$ and $\mathbf{x}^2$, if $\mathbf{x}^1 \preceq_{(\mathbf{b},A)} \mathbf{x}^2$, and $x_j^1 = x_j^2 = 0$, then for any $\hat{\mathbf{a}}$

$$\mathbf{x}^1 \preceq_{(\mathbf{b},\hat{A})} \mathbf{x}^2, \quad \text{where } \hat{A} = (\mathbf{a}_1, \cdots, \mathbf{a}_{j-1}, \hat{\mathbf{a}}, \mathbf{a}_{j+1}, \cdots, \mathbf{a}_m)\,.$$

(ii) Given $\mathbf{b}$, $A$, $\mathbf{x}^1$, $\mathbf{x}^2$, $b'$ and $\mathbf{z}$, if $\mathbf{x}^1 \preceq_{(\mathbf{b},A)} \mathbf{x}^2$, $\quad b = \mathbf{z}^\top \mathbf{x}^2$ then

$$\mathbf{x}^1 \preceq_{(\overline{\mathbf{b}},\overline{A})} \mathbf{x}^2, \quad \text{where } \overline{\mathbf{b}} = \begin{pmatrix} \mathbf{b} \\ b' \end{pmatrix}; \quad \overline{A} = \begin{pmatrix} A \\ \mathbf{z}^\top \end{pmatrix}.$$

(iii) Given $j$, $\mathbf{b}$, $A$, $\mathbf{x}^1$ and $\mathbf{x}^2$, if $\mathbf{x}^1 \preceq_{(\mathbf{b},A)} \mathbf{x}^2$, then

$$\hat{\mathbf{x}}^1 \preceq_{(\mathbf{b},\tilde{A})} \hat{\mathbf{x}}^2, \quad \text{where } \hat{\mathbf{x}}^i = \begin{pmatrix} \mathbf{x}^i \\ 0 \end{pmatrix},\ i = 1, 2; \quad \tilde{A} = (A, \mathbf{0})\,.$$

(iv) Given $\mathbf{b}$, $A$, $\mathbf{x}^1$,$\mathbf{x}^2$ and $P \in \mathbb{R}^{m \times m}$ a permutation matrix, if $\mathbf{x}^1 \preceq_{(\mathbf{b},A)} \mathbf{x}^2$, then $P^\top \mathbf{x}^1 \preceq_{(\mathbf{b},AP)} P^\top \mathbf{x}^2$.

## General proof

- Choose $(\mathbf{b}, A)$ and $(0, \mathbf{z}^\top)$ to jointly achieve $\mathfrak{b}_n(\mathbb{L}, \mathcal{Z})$: for some $\mathbf{x}^* \in \mathbb{L}(\mathbf{b}, A)$, $\mathfrak{b}_n(\mathbb{L}, \mathcal{Z}) = l\Big(\mathbf{x}^*, (0, \mathbf{z})\Big)$.

## General proof

- Choose $(\mathbf{b}, A)$ and $(0, \mathbf{z}^\top)$ to jointly achieve $\mathfrak{b}_n(\mathbb{L}, \mathcal{Z})$: for some $\mathbf{x}^* \in \mathbb{L}(\mathbf{b}, A)$, $\mathfrak{b}_n(\mathbb{L}, \mathcal{Z}) = l\Big(\mathbf{x}^*, (0, \mathbf{z})\Big)$.
- Let

$$\hat{\mathbf{z}} \triangleq (\mathbf{0}^\top, \mathbf{z}^\top); \qquad \hat{A} \triangleq (A, A);$$

$$\tilde{\mathbf{b}} \triangleq \begin{pmatrix} \mathbf{b} \\ 0 \end{pmatrix}; \qquad \tilde{A} \triangleq \begin{pmatrix} A, & A \\ \mathbf{0}^\top, & \mathbf{z}^\top \end{pmatrix}.$$

## General proof

- Choose $(\mathbf{b}, A)$ and $(0, \mathbf{z}^\top)$ to jointly achieve $\mathfrak{b}_n(\mathbb{L}, \mathcal{Z})$: for some $\mathbf{x}^* \in \mathbb{L}(\mathbf{b}, A)$, $\mathfrak{b}_n(\mathbb{L}, \mathcal{Z}) = l\Big(\mathbf{x}^*, (0, \mathbf{z})\Big)$.
- Let

$$\hat{\mathbf{z}} \triangleq (\mathbf{0}^\top, \mathbf{z}^\top); \qquad \hat{A} \triangleq (A, A);$$
$$\tilde{\mathbf{b}} \triangleq \begin{pmatrix} \mathbf{b} \\ 0 \end{pmatrix}; \qquad \tilde{A} \triangleq \begin{pmatrix} A, & A \\ \mathbf{0}^\top, & \mathbf{z}^\top \end{pmatrix}.$$

- By assumptions on partial ordering *and sparsity of algorithm* $\mathbb{L}$:

$$\begin{pmatrix} \mathbf{0} \\ \mathbf{x}^* \end{pmatrix} \in \mathbb{L}_{(\mathbf{b}, \hat{A})}; \quad \begin{pmatrix} \mathbf{x}^* \\ \mathbf{0} \end{pmatrix} \in \mathbb{L}_{(\tilde{\mathbf{b}}, \tilde{A})}.$$

## General proof

- Choose $(\mathbf{b}, A)$ and $(0, \mathbf{z}^\top)$ to jointly achieve $\mathfrak{b}_n(\mathbb{L}, \mathcal{Z})$: for some $\mathbf{x}^* \in \mathbb{L}(\mathbf{b}, A)$, $\mathfrak{b}_n(\mathbb{L}, \mathcal{Z}) = l\Big(\mathbf{x}^*, (0, \mathbf{z})\Big)$.
- Let

$$\hat{\mathbf{z}} \triangleq (\mathbf{0}^\top, \mathbf{z}^\top); \qquad \hat{A} \triangleq (A, A);$$
$$\tilde{\mathbf{b}} \triangleq \begin{pmatrix} \mathbf{b} \\ 0 \end{pmatrix}; \qquad \tilde{A} \triangleq \begin{pmatrix} A, & A \\ \mathbf{0}^\top, & \mathbf{z}^\top \end{pmatrix}.$$

- By assumptions on partial ordering *and sparsity of algorithm* $\mathbb{L}$:

$$\begin{pmatrix} \mathbf{0} \\ \mathbf{x}^* \end{pmatrix} \in \mathbb{L}_{(\mathbf{b}, \hat{A})}; \quad \begin{pmatrix} \mathbf{x}^* \\ \mathbf{0} \end{pmatrix} \in \mathbb{L}_{(\tilde{\mathbf{b}}, \tilde{A})}.$$

- This leads to: $l\Big(\mathbb{L}_{(\mathbf{b}, \hat{A})}, (0, \hat{\mathbf{z}})\Big) = l(\mathbf{x}^*, (0, \mathbf{z})); \quad l\Big(\mathbb{L}_{(\tilde{\mathbf{b}}, \tilde{A})}, (0, \hat{\mathbf{z}})\Big) = 0.$

## General proof

- Choose $(\mathbf{b}, A)$ and $(0, \mathbf{z}^\top)$ to jointly achieve $\mathfrak{b}_n(\mathbb{L}, \mathcal{Z})$: for some $\mathbf{x}^* \in \mathbb{L}(\mathbf{b}, A)$, $\mathfrak{b}_n(\mathbb{L}, \mathcal{Z}) = l\Big(\mathbf{x}^*, (0, \mathbf{z})\Big)$.
- Let

$$\hat{\mathbf{z}} \triangleq (\mathbf{0}^\top, \mathbf{z}^\top); \qquad \hat{A} \triangleq (A,\ A);$$
$$\tilde{\mathbf{b}} \triangleq \begin{pmatrix} \mathbf{b} \\ 0 \end{pmatrix}; \qquad \tilde{A} \triangleq \begin{pmatrix} A, & A \\ \mathbf{0}^\top, & \mathbf{z}^\top \end{pmatrix}.$$

- By assumptions on partial ordering *and sparsity of algorithm* $\mathbb{L}$:

$$\begin{pmatrix} \mathbf{0} \\ \mathbf{x}^* \end{pmatrix} \in \mathbb{L}_{(\mathbf{b}, \hat{A})}; \quad \begin{pmatrix} \mathbf{x}^* \\ \mathbf{0} \end{pmatrix} \in \mathbb{L}_{(\tilde{\mathbf{b}}, \tilde{A})}.$$

- This leads to: $l\Big(\mathbb{L}_{(\mathbf{b}, \hat{A})}, (0, \hat{\mathbf{z}})\Big) = l(\mathbf{x}^*, (0, \mathbf{z})); \quad l\Big(\mathbb{L}_{(\tilde{\mathbf{b}}, \tilde{A})}, (0, \hat{\mathbf{z}})\Big) = 0.$
- Hence we have:

$$\beta \geq \mathfrak{b}_n(\mathbb{L}, \mathcal{Z}) = l\Big(\mathbb{L}_{(\mathbf{b}, \hat{A})}, (0, \hat{\mathbf{z}})\Big) - l\Big(\mathbb{L}_{(\tilde{\mathbf{b}}, \tilde{A})}, (0, \hat{\mathbf{z}})\Big).$$

## Some other thoughts

- Of course: consistency does not require stability. Other ways to prove PAC bounds/consistency?
- Non-uniform stability?
- Other price to pay for sparsity?
- Other problems: classification, SVMs.

## Some other thoughts

- Of course: consistency does not require stability. Other ways to prove PAC bounds/consistency?
- Non-uniform stability?
- Other price to pay for sparsity?
- Other problems: classification, SVMs.

mpexuh@nus.edu.sg
http://guppy.mpe.nus.edu.sg/ mpexuh/