Part II of today's talk

- Structure Learning of Discrete Pairwise Graphical Models
  - Sparse logistic regression provably recovers the graph structure
    - Sample complexity improves the previous state-of-the-art (k^5 vs k^4)
    - Can be efficiently optimized (total runtime $\tilde{O}(n^2)$)
  - Experimental results support our analysis
- Learning a Compressed Sensing Measurement Matrix

Motivation

- High-dimensional data are often sparse
  - Amazon employee dataset: $d = 15k$, nnz = 9
  - RCV1 text dataset: $d = 47k$, nnz = 76
  - Wiki multi-label dataset: $d = 31k$, nnz = 19

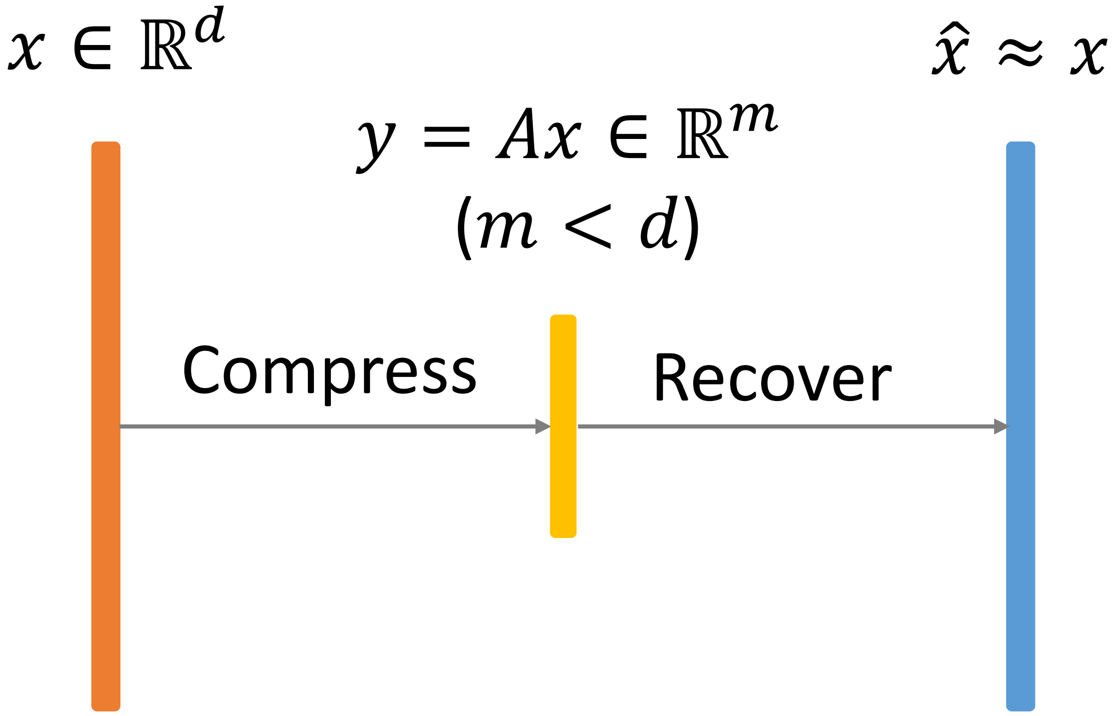
One-hot encoded categorical data

- Unlike image/video data, there is no notion of spatial/time locality

- Reduce the dimensionality via a linear sketching/embedding

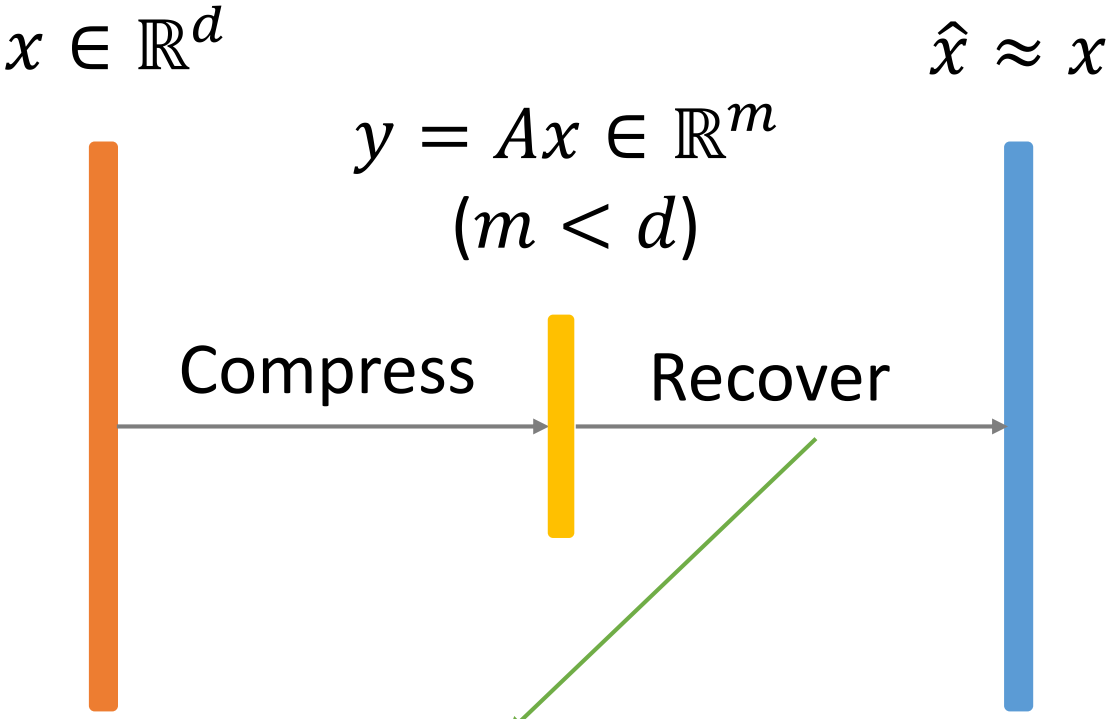
Can we have a **lossless** linear sketch of high-dimensional sparse data?

Compressed Sensing (Donoho; Candès et al.; ...)



- Suppose x is sparse
- $A \in \mathbb{R}^{m \times d}$ satisfies RIP
- Recovery algorithms
  - ℓ_1-min, CoSaMP, IHT, AMP...
- NP-hard to check RIP
  - Gaussian matrix works w.h.p.

Compressed Sensing (Donoho; Candès et al.; ...)



ℓ_1-min (aka basis pursuit):

$$\hat{x} := \operatorname{argmin}_{x'} \|x'\|_1 \quad \text{s.t. } Ax' = y$$

- Suppose x is sparse
- $A \in \mathbb{R}^{m \times d}$ satisfies RIP
- Recovery algorithms
  - ℓ_1-min, CoSaMP, IHT, AMP...
- NP-hard to check RIP
  - Gaussian matrix works w.h.p.

Model-based Compressed Sensing (Baraniuk et al.)

- If in addition to sparsity, x satisfies a known sparsity model $\mathbb{M}$
  - E.g., $\mathbb{M}$ is the block-sparsity model or tree-sparsity model

- Then model-based CoSaMP is better than vanilla CoSaMP

Project onto $\mathbb{M}$ in every iteration

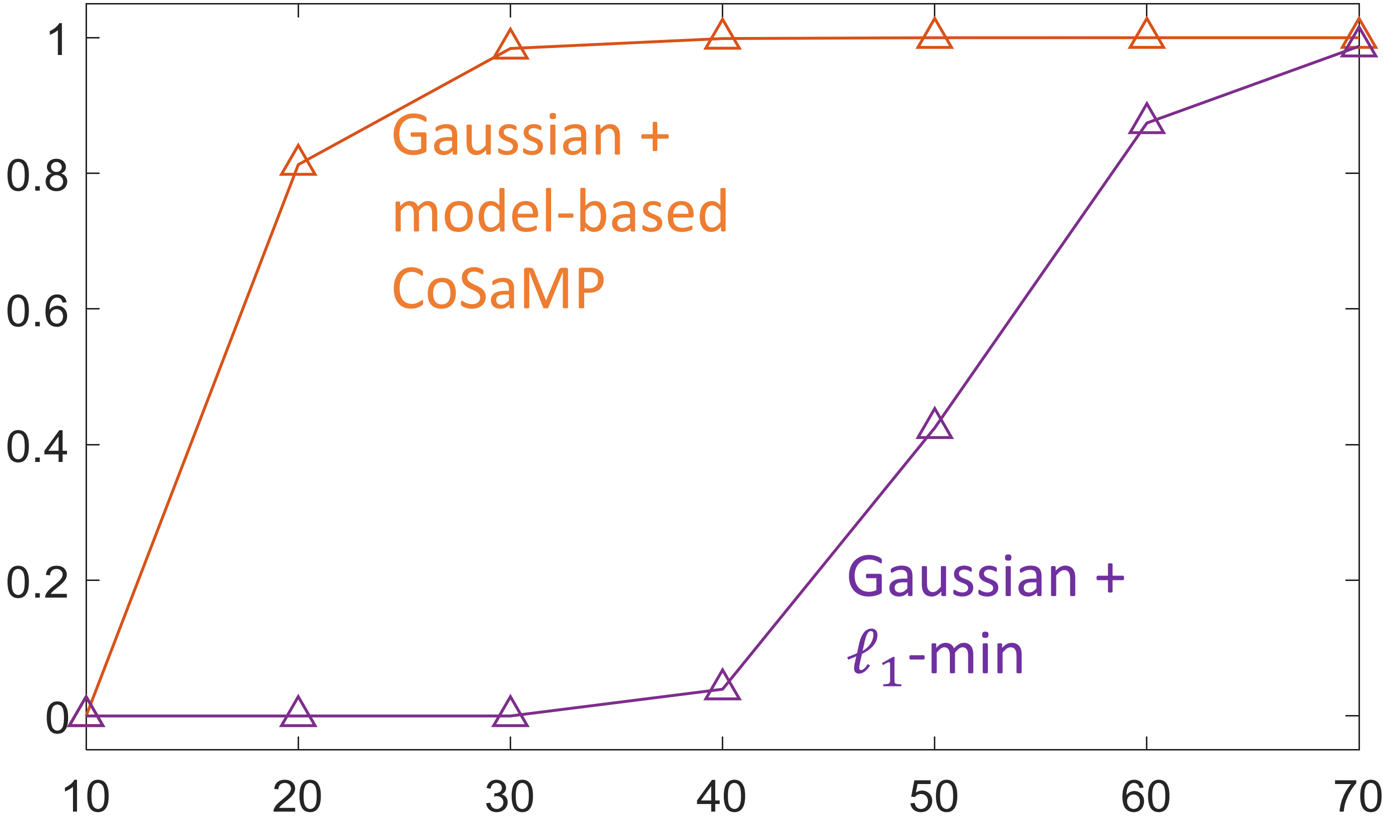
Let's look at an example

- Generate 1000 random sparse vectors in $\mathbb{R}^{1000}$
  - Each vector has 10 non-zeros
  - Support ∈ {1-10, 11-20, …, 991-1000} Block sparse
  - Set the non-zeros as Uniform[0,1]
- Measurement matrix:
  - Random Gaussian matrix
- Recovery algorithms:
  - ℓ_1-min: $\hat{x} \coloneqq \mathrm{argmin}_{x'} \|x'\|_1 \quad \mathrm{s.t.}\ Ax' = y$
  - Model-based CoSaMP: block-sparsity model

Comparisons of the recovery performance

Fraction of exactly recovered points

Exact recovery: $\|x - \hat{x}\|_2 \leq 10^{-10}$

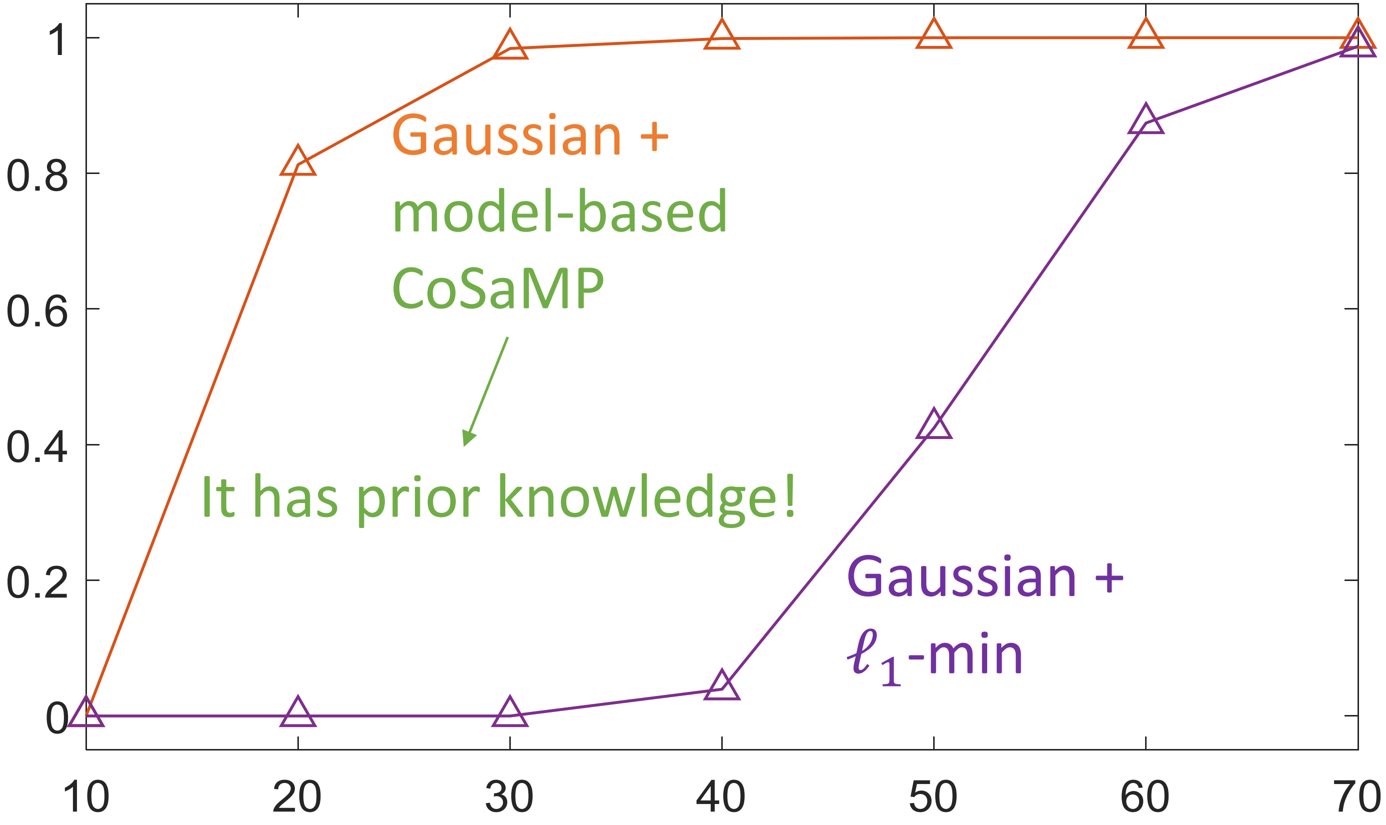


Number of measurements (m)

Comparisons of the recovery performance

Fraction of exactly recovered points

Exact recovery:
$\|x - \hat{x}\|_2 \leq 10^{-10}$



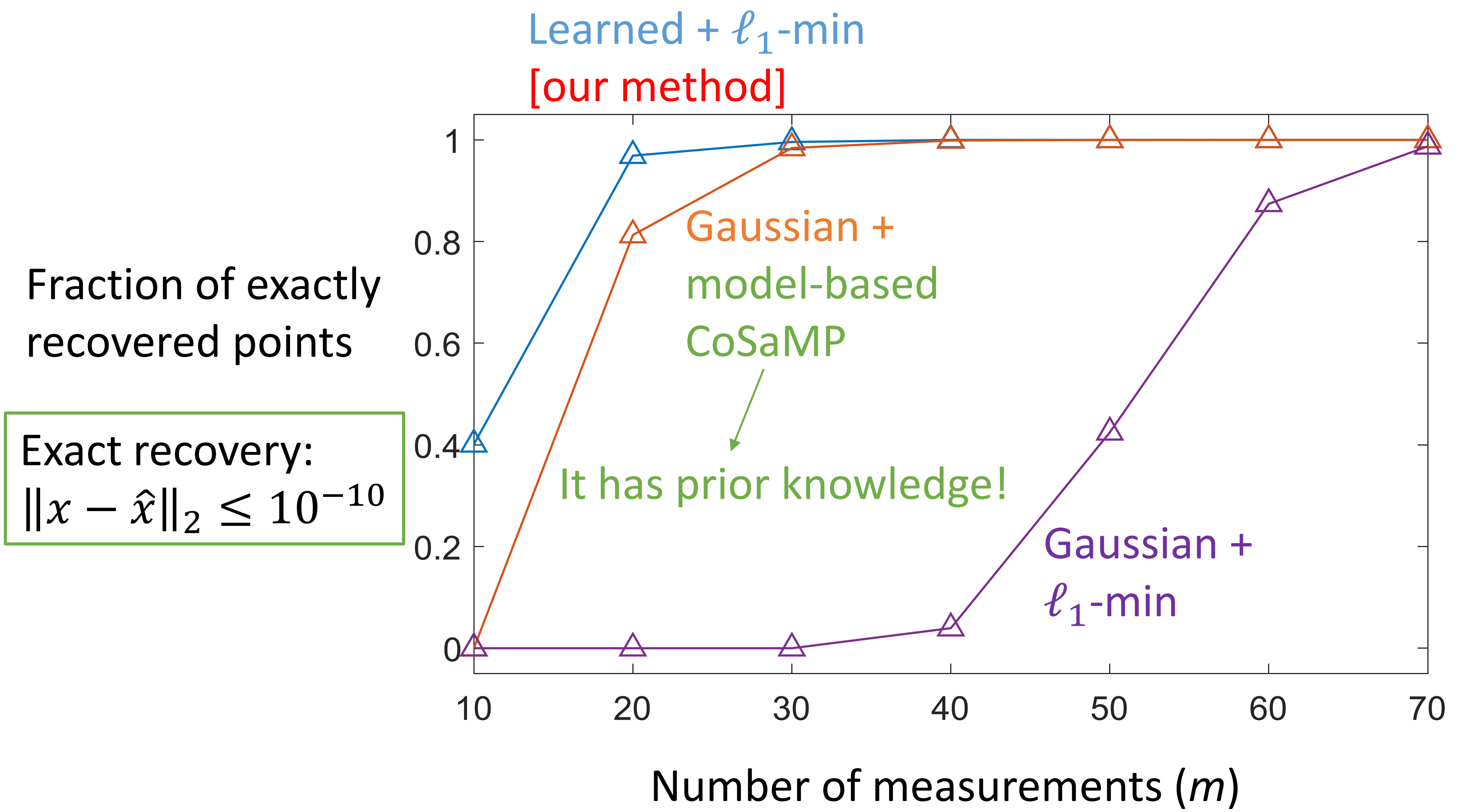
Number of measurements (m)

What if we do not have the prior knowledge?

Given n sparse vectors $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ as training samples.

Can we still achieve performance similar to model-based CoSaMP?

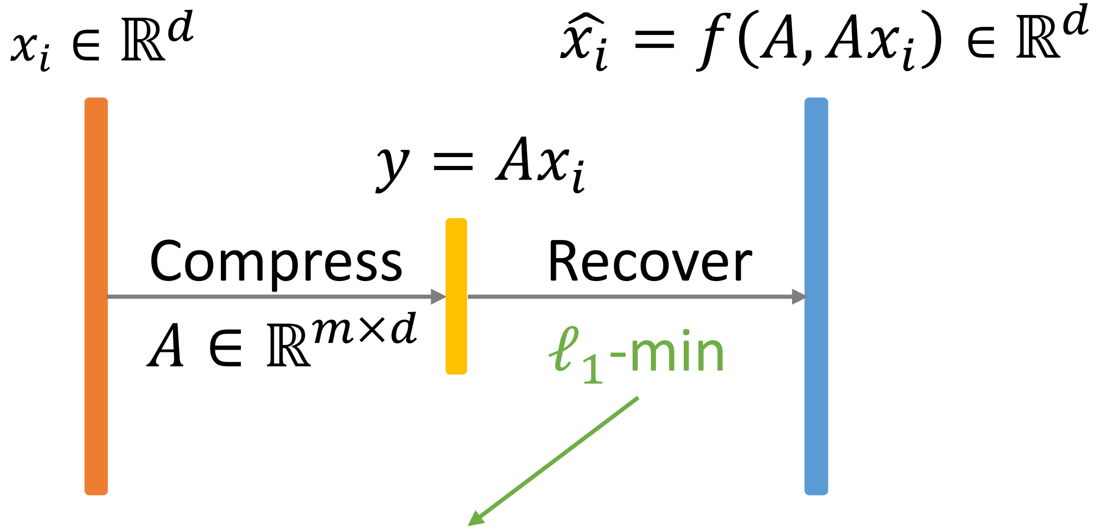
Comparisons of the recovery performance



Exact recovery:
$\|x - \hat{x}\|_2 \leq 10^{-10}$

Learning a measurement matrix

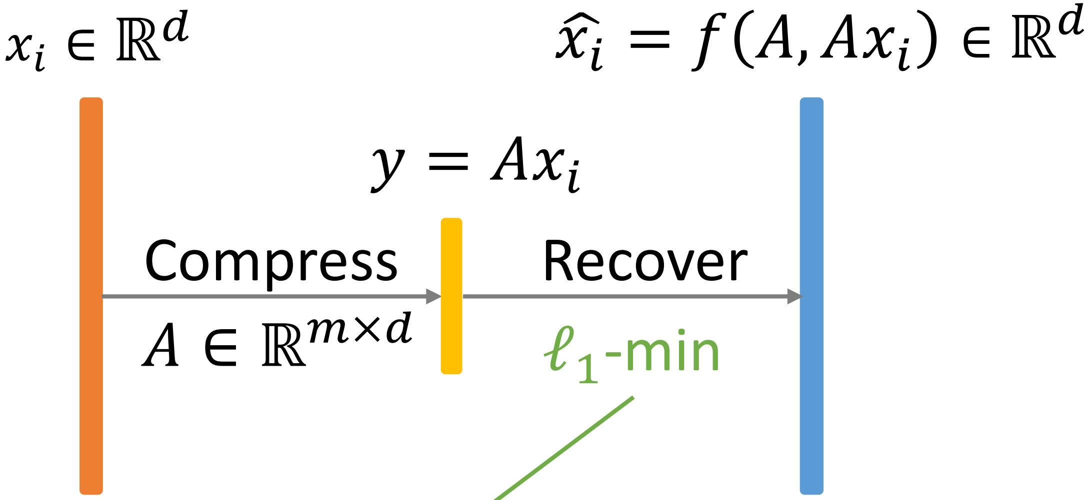
- Training data: n sparse vectors $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$



$$f(A, y) := \operatorname{argmin}_{x'} \|x'\|_1 \quad \text{s.t. } Ax' = y$$

Learning a measurement matrix

- Training data: n sparse vectors $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$



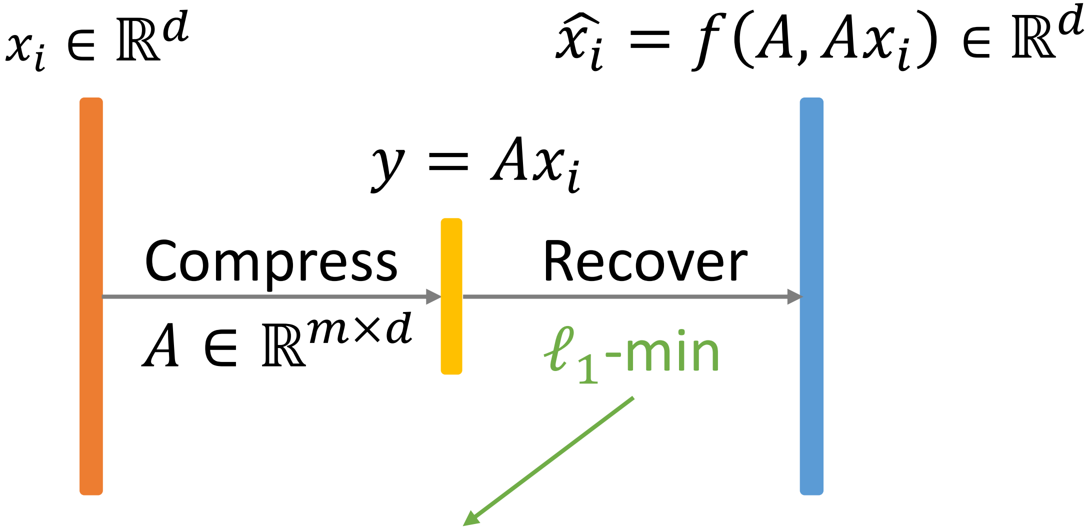
$$f(A, y) := \operatorname{argmin}_{x'} \|x'\|_1 \quad \text{s.t. } Ax' = y$$

Objective function:

$$\min_{A \in \mathbb{R}^{m \times d}} \sum_{i=1}^{n} \|x_i - f(A, Ax_i)\|_2^2$$

Learning a measurement matrix

- Training data: n sparse vectors $x_1, x_2, \dots, x_n \in \mathbb{R}^d$



$$f(A, y) \coloneqq \operatorname{argmin}_{x'} \|x'\|_1 \quad \text{s.t. } Ax' = y$$

Objective function:

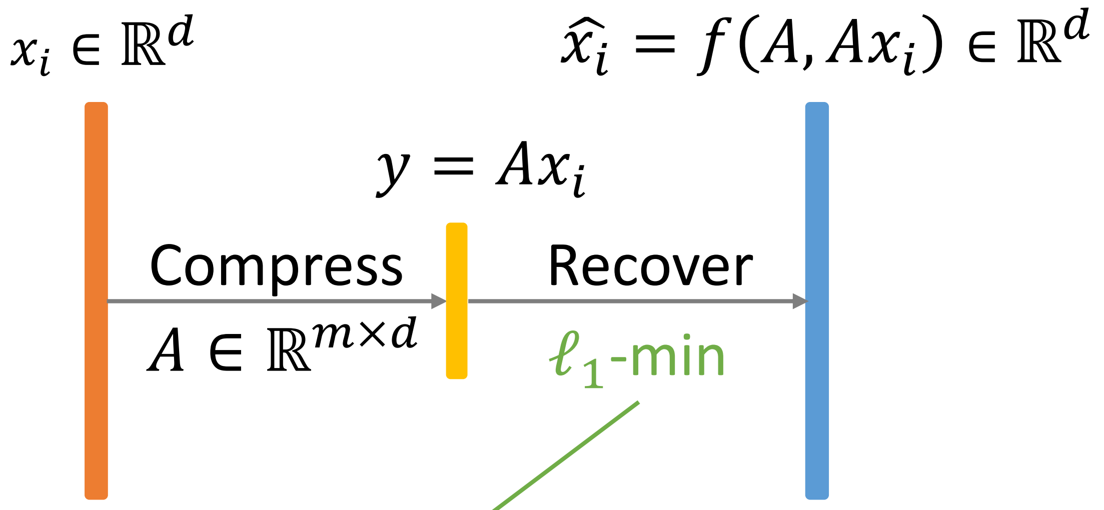
$$\min_{A \in \mathbb{R}^{m \times d}} \sum_{i=1}^{n} \|x_i - f(A, Ax_i)\|_2^2$$

Problem:

How to compute gradient w.r.t. A?

Learning a measurement matrix

- Training data: n sparse vectors $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$



$$f(A, y) \coloneqq \operatorname{argmin}_{x'} \|x'\|_1 \quad \text{s.t. } Ax' = y$$

Objective function:

$$\min_{A \in \mathbb{R}^{m \times d}} \sum_{i=1}^{n} \|x_i - f(A, Ax_i)\|_2^2$$

Problem:

How to compute gradient w.r.t. A?

Key idea:

Replace $f(A, y)$ by a few steps of projected subgradient

Projected subgradient of ℓ_1-min

$$f(A, y) \coloneqq \operatorname{argmin}_{x'} \|x'\|_1 \quad \text{s.t. } Ax' = y$$

- **Initialize**: $x^{(1)} = A^\dagger y$
- **Iterate**: for *t = 1, 2, …, T* do

$$x^{(t+1)} = \Pi(x^{(t)} - \alpha_t \operatorname{sign}(x^{(t)}))$$
$$= x^{(t)} - \alpha_t (I - A^\dagger A) \operatorname{sign}(x^{(t)})$$

- **Define:** $\widetilde{f}_T(A, y) \coloneqq x^{(T+1)}$

$A^\dagger = A^T (AA^T)^{-1}$: Pseudoinverse

$Ax^{(1)} = y$

Π: projection onto $\{x : Ax = y\}$

α_t: step size at t-th iteration

Projected subgradient of ℓ_1-min

$$f(A, y) \coloneqq \operatorname{argmin}_{x'} \|x'\|_1 \quad \text{s.t. } Ax' = y$$

- **Initialize**: $x^{(1)} = A^{\dagger} y$
- **Iterate**: for t = 1, 2, …, T do

$$x^{(t+1)} = \Pi\left(x^{(t)} - \alpha_t \operatorname{sign}(x^{(t)})\right)$$

$$= x^{(t)} - \alpha_t \left(I - A^{\dagger} A\right) \operatorname{sign}(x^{(t)})$$

- **Define**: $\widetilde{f}_T(A, y) \coloneqq x^{(T+1)}$

$A^{\dagger} = A^T (AA^T)^{-1}$: Pseudoinverse

$Ax^{(1)} = y$

Π: projection onto $\{x : Ax = y\}$

α_t: step size at t-th iteration

Projected subgradient of ℓ_1-min

$$f(A, y) \coloneqq \operatorname{argmin}_{x'} \|x'\|_1 \quad \text{s.t. } Ax' = y$$

- **Initialize**: $x^{(1)} = A^{\dagger} y$
- **Iterate**: for $t = 1, 2, \ldots, T$ do

$$x^{(t+1)} = \Pi\left(x^{(t)} - \alpha_t \operatorname{sign}(x^{(t)})\right)$$
$$= x^{(t)} - \alpha_t \left(I - A^{\dagger} A\right) \operatorname{sign}(x^{(t)})$$

- **Define**: $\tilde{f}_T(A, y) \coloneqq x^{(T+1)}$

$A^{\dagger} = A^T (AA^T)^{-1}$: Pseudoinverse

$Ax^{(1)} = y$

Π: projection onto $\{x : Ax = y\}$

α_t: step size at t-th iteration

Projected subgradient of ℓ_1-min

$$f(A, y) := \operatorname{argmin}_{x'} \|x'\|_1 \quad \text{s.t. } Ax' = y$$

- **Initialize**: $x^{(1)} = A^{\dagger} y$
- **Iterate**: for $t = 1, 2, \ldots, T$ do

$$x^{(t+1)} = \Pi\left(x^{(t)} - \alpha_t \operatorname{sign}(x^{(t)})\right)$$
$$= x^{(t)} - \alpha_t \left(I - A^{\dagger} A\right) \operatorname{sign}(x^{(t)})$$

- **Define:** $\tilde{f}_T(A, y) := x^{(T+1)}$

$A^{\dagger} = A^T (AA^T)^{-1}$: Pseudoinverse

$Ax^{(1)} = y$

Π: projection onto $\{x : Ax = y\}$

α_t: step size at t-th iteration

Summary of our idea

Objective function:

$$\min_{A \in \mathbb{R}^{m\times d}} \sum_{i=1}^{n} \|x_i - f(A, Ax_i)\|_2^2$$

Problem:

Hard to compute gradient w.r.t. A

Key idea:

Replace $f(A, y)$ by $\tilde{f}_T(A, y)$,
where $\tilde{f}_T(A, y) \coloneqq T$ steps of
projected subgradient update

New objective function:

$$\min_{A \in \mathbb{R}^{m\times d}} \sum_{i=1}^{n} \left\|x_i - \tilde{f}_T(A, Ax_i)\right\|_2^2$$

Summary of our idea

Objective function:

$$\min_{A \in \mathbb{R}^{m \times d}} \sum_{i=1}^{n} \|x_i - f(A, Ax_i)\|_2^2$$

Problem:

Hard to compute gradient w.r.t. A

Key idea:

Replace $f(A, y)$ by $\tilde{f}_T(A, y)$, where $\tilde{f}_T(A, y) :=$ T steps of projected subgradient update

$$\frac{\partial \tilde{f}_T}{\partial A} \rightarrow \frac{\partial \tilde{f}_{T-1}}{\partial A} \rightarrow \cdots\cdots \rightarrow \frac{\partial \tilde{f}_0}{\partial A}$$

New objective function:

$$\min_{A \in \mathbb{R}^{m \times d}} \sum_{i=1}^{n} \left\|x_i - \tilde{f}_T(A, Ax_i)\right\|_2^2$$

Summary of our idea

Objective function:

$$\min_{A \in \mathbb{R}^{m\times d}} \sum_{i=1}^{n} \|x_i - f(A, Ax_i)\|_2^2$$

Problem:

Hard to compute gradient w.r.t. A

Key idea:

Replace $f(A, y)$ by $\tilde{f}_T(A, y)$, where $\tilde{f}_T(A, y) :=$ T steps of projected subgradient update

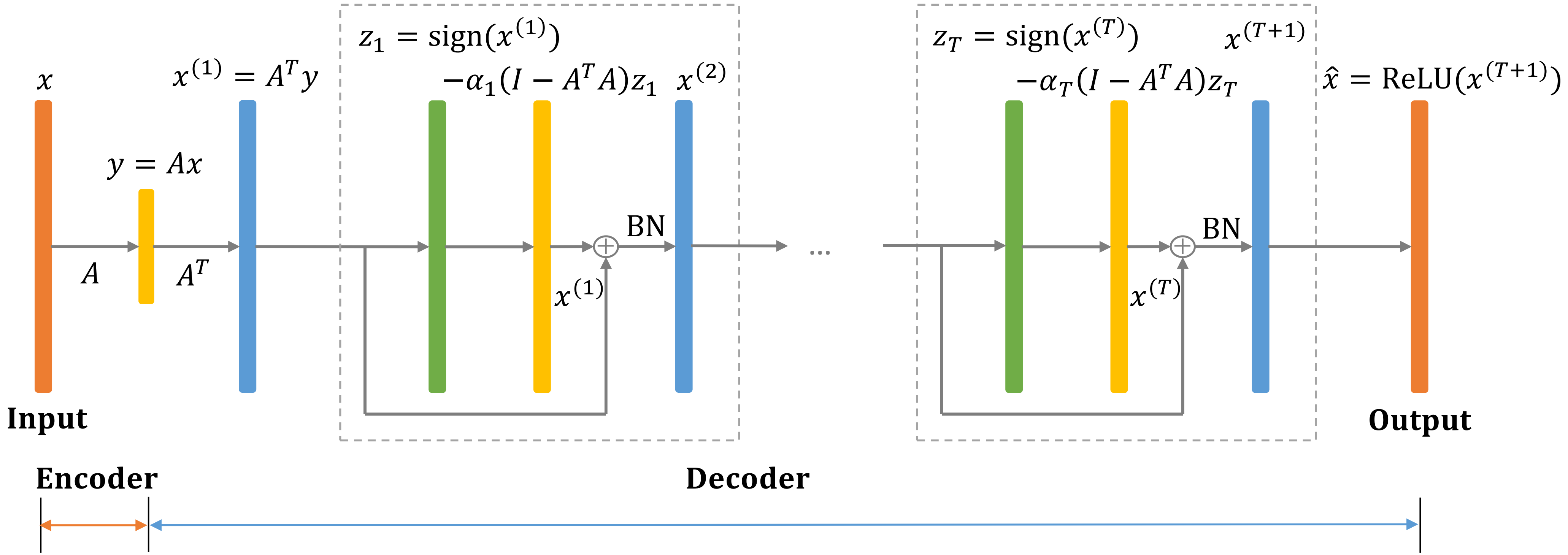
$$\frac{\partial \tilde{f}_T}{\partial A} \rightarrow \frac{\partial \tilde{f}_{T-1}}{\partial A} \rightarrow \cdots\cdots \rightarrow \frac{\partial \tilde{f}_0}{\partial A}$$

New objective function:

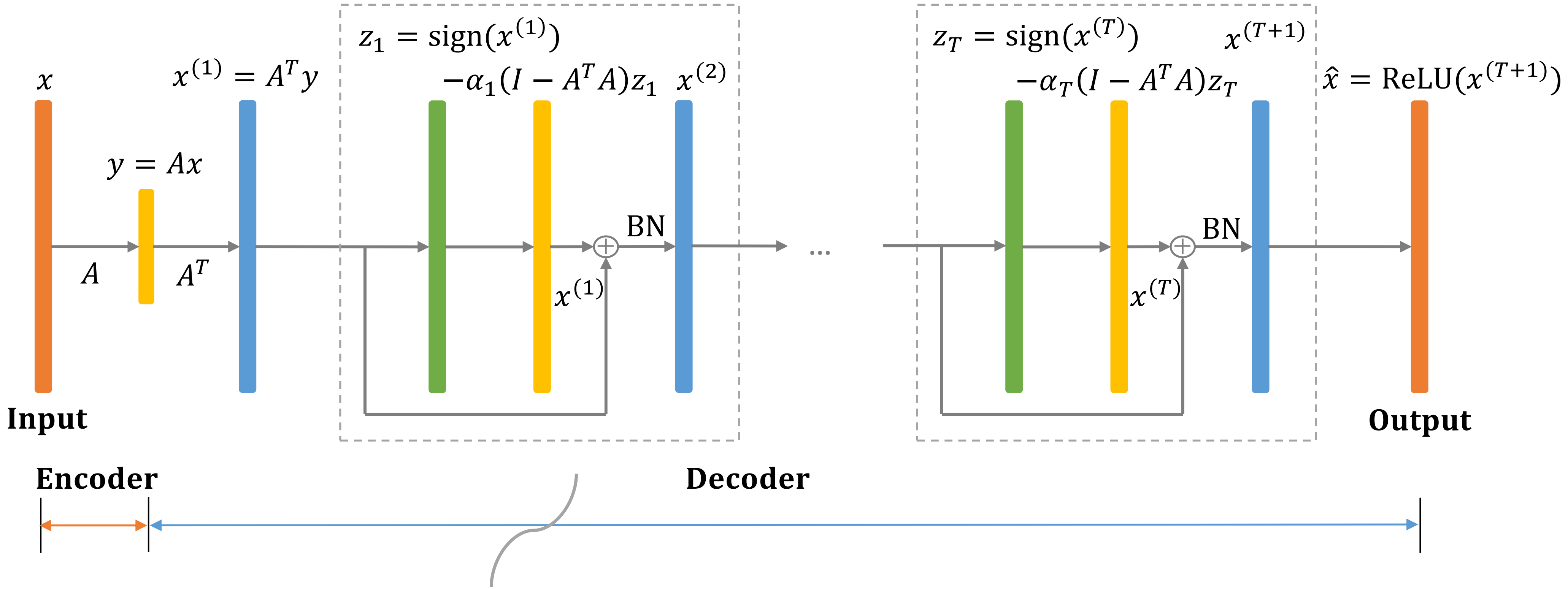
$$\min_{A \in \mathbb{R}^{m\times d}} \sum_{i=1}^{n} \left\|x_i - \tilde{f}_T(A, Ax_i)\right\|_2^2$$

Tuning param: T=10 in the experiments

ℓ_1-AE: a novel autoencoder architecture



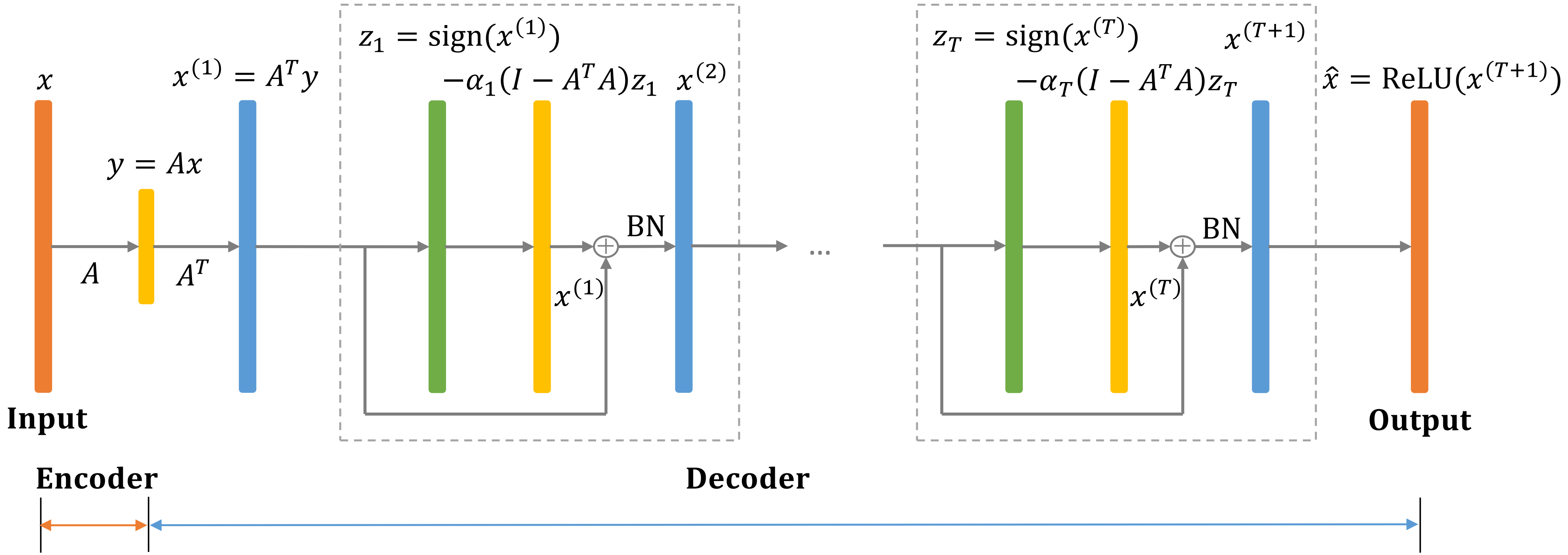
ℓ_1-AE: a novel autoencoder architecture



One step of projected subgradient

$$x^{(t+1)} = x^{(t)} - \alpha_t(I - A^T A)\text{sign}(x^{(t)})$$

ℓ_1-AE: a novel autoencoder architecture



Training objective function:

$$\min_{A \in \mathbb{R}^{m \times d}, \{\alpha_t\}_{t=1}^T} \frac{1}{n} \sum_{i=1}^n \|x_i - \hat{x}_i(A, \{\alpha_t\}_{t=1}^T)\|_2^2$$

Performance metrics

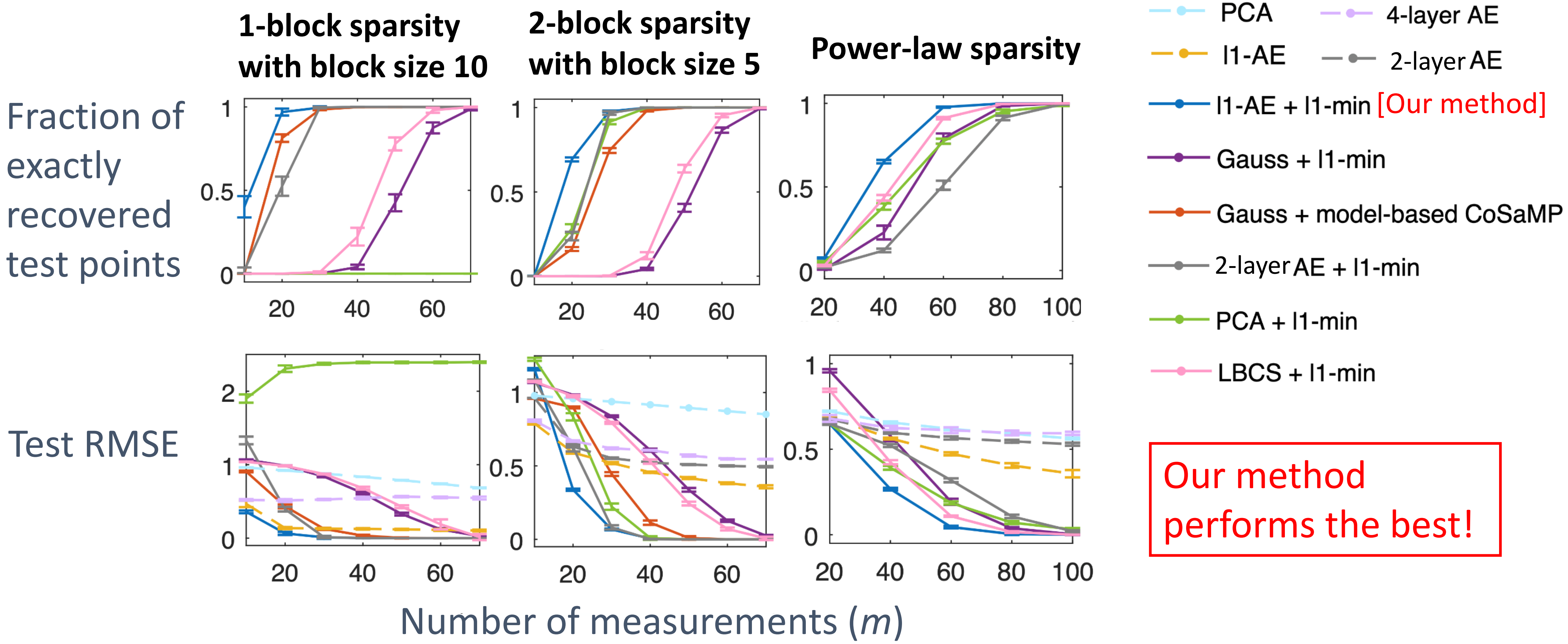
- We evaluate two performance metrics on the test data set.
- Metric 1: Fraction of exactly recovered points

$$\underbrace{\text{Fraction of exactly recovered points}}_{\|x_i - \widehat{x_i}\| \leq 10^{-10}}$$

- Metric 2: Test RMSE

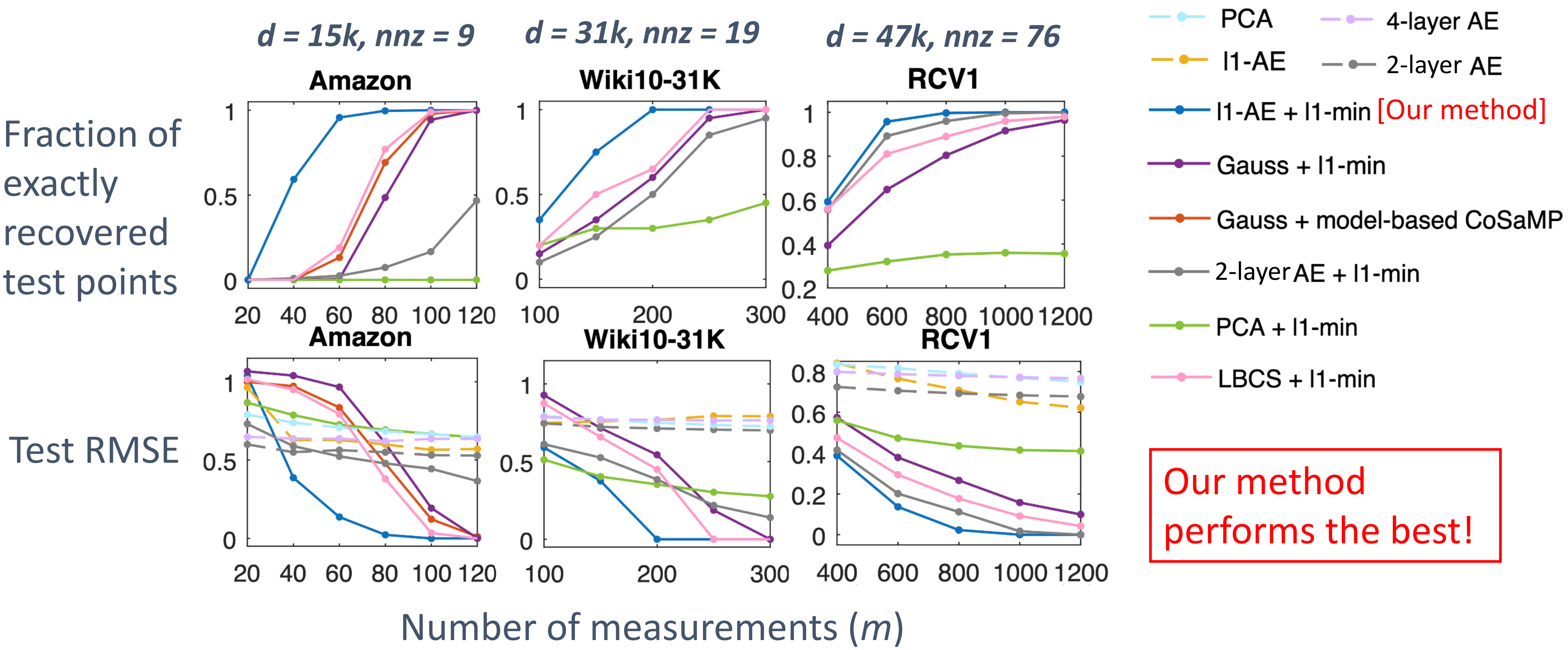
$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\|x_i - \widehat{x_i}\|_2^2}$$

Synthetic data: d = 1k, nnz = 10



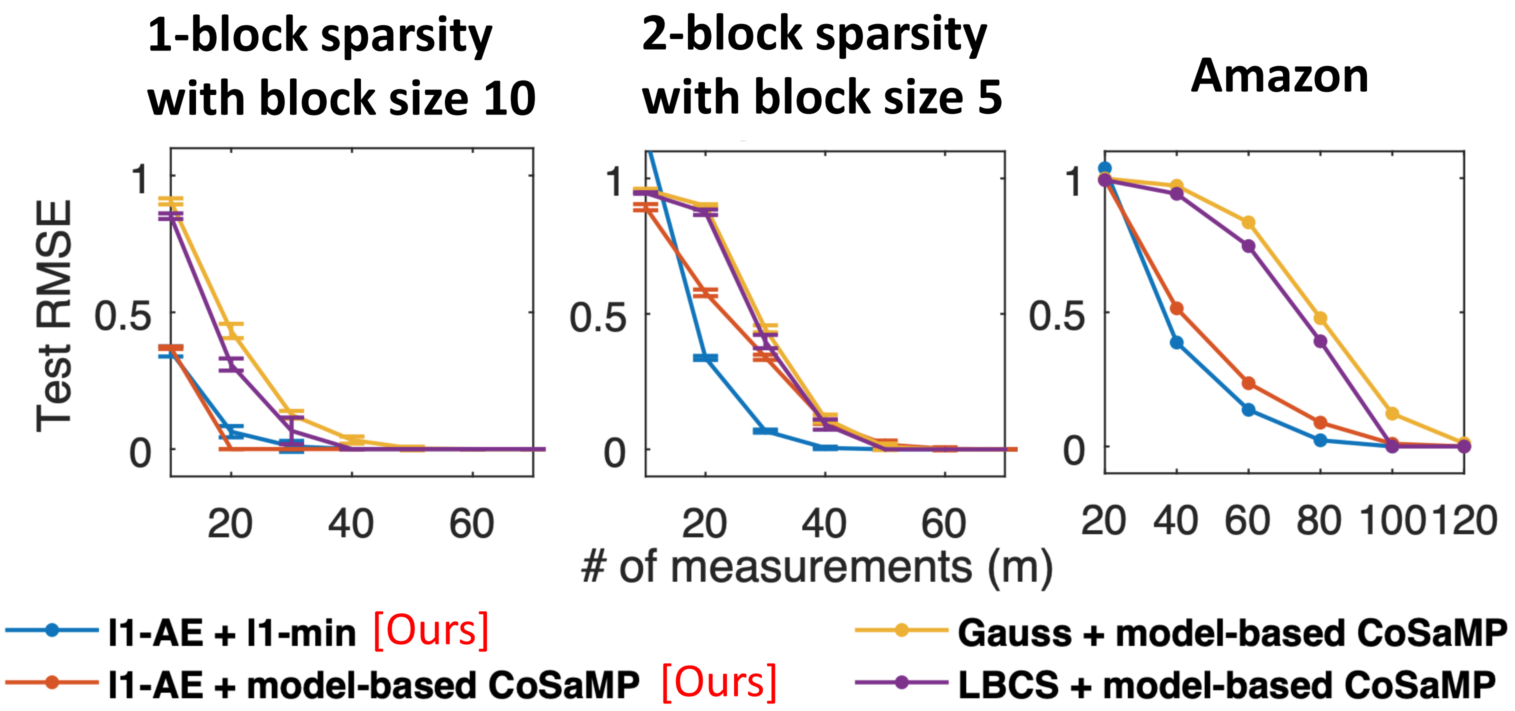
Our method performs the best!

Real-world sparse datasets



Our method performs the best!

ℓ_1-AE + model-based decoder



ℓ_1-AE also improves the performance of model-based CoSaMP

Unrolling/unfolding an iterative algorithm

- Sparse coding [Gregor and LeCun'2010, …]
- NMF [Hershey et al.'2014]
- Image reconstruction [Mardani et al.'2017, …]
- Optimization [Andrychowicz et al.'2016, …]
- Adversarial attacks [Zugner and Gunnemann'2019, …]
- Learning a compressed sensing matrix for ℓ_1-min [Ours]

Seek a trained NN to replace the original iterative algorithm in inference

Use the learned matrix in the original algorithm

Summary of Part II

- Structure Learning of Discrete Pairwise Graphical Models
  - Sparse logistic regression provably recovers the graph structure
    - Sample complexity improves the previous state-of-the-art (k^5 vs k^4)
    - Can be efficiently optimized (total runtime $\tilde{O}(n^2)$)
  - Experimental results support our analysis
- Learning a Compressed Sensing Measurement Matrix
  - By unrolling the projected subgradient of ℓ_1-min
    - Implemented as an autoencoder ℓ_1-AE
  - Compare 12 algorithms over 6 datasets (3 synthetic and 3 real)
    - Our method can create perfect reconstruction with 1.1-3X fewer measurements than the state-of-the-art methods