Problem Setup

Premise: To devise a sample-efficient linear-convergence algorithm for phase retrieval of s-sparse signals with underlying structured sparsity patterns.

Main Challenges

- Linear-convergence algorithms have sample complexity with quadratic dependence on sparsity \( m = \mathcal{O}(s^2 \log n) \).
- High number of tuning parameters.
- Lower sample complexity algorithms require high run time, not scalable.

Prior Work

- Convex: PhaseLift, PhaseMax.

Drawbacks:

- Structured sparsity models: Designed sensing matrices.
- Requires fast and scalable.
- Lower sample complexity algorithms require high run time, not scalable.

Our Objective

Inefficient phase retrieval algorithms for structured signals.

- Is naturally compatible with standard sparse recovery algorithms.
- Is fast and scalable to large datasets of large dimensions.
- Has sub-quadratic sample complexity \( m = \mathcal{O}\left(\frac{s}{1} \log n\right)\).
- Requires no extra parametric inputs apart from (block) sparsity \( k = \frac{s}{1} \).

New Direction: Phase retrieval of structured sparse signals

Problem Setup: Recover signal \( x^0 \in \mathbb{R}^n \), using Gaussian sampling matrix \( A = [a_1, \ldots, a_m] \), from measurements \( y \in \mathbb{R}^m \).

Ideas: Efficient phase retrieval algorithms for structured sparse signals.

Solution Methodology

Formulate the above as a bi-convex problem, by introducing diagonal phase matrix \( P \in \mathbb{P} \) with \( P_{ij} = \text{sign}(a_i^T x) \in \{1, -1\} \), and alternatively minimize the loss function over variables \( x \) and \( P \):

\[
\min_{x, \mathcal{P} \in \mathbb{P}} \|Ax - Py\|_2.
\]

This strategy requires a good initial point, to converge to minimum. For this, we introduce our algorithm Block Compressive Phase Retrieval with Alternating Minimization (Block CoPRAM) [JH17].

Block CoPRAM - Smart Initialization

Setup: Define signal marginals as \( M_j = \frac{1}{m} \sum_{i=1}^{m} y_i^2 \phi_j^a \) for \( j \in \{1, \ldots n\} \).

Objective: Find good initial estimate \( x^0 \) of the true signal \( x^* \).

Challenge: Designing block marginals. Performance guarantees.

Solution: (Alg. 1)

- Define block marginals as \( M_{jk} = \sqrt{\sum_{j=1}^{k} M_j} \), for \( j = \{1, \ldots n\} \).
- Reject bottom \( (n_k - k) \) block marginals: \( \hat{S}, \text{card}(\hat{S}) = k \).
- Construct matrix \( (\in \mathbb{R}^{nk \times n}) M_k = \frac{1}{m} \sum_{i=1}^{m} y_i^2 \phi_j^a \).
- Top left-singular vector \( \phi_m \) of \( M_k \): initial estimate \( x^0 = \phi_m \), where \( \phi_m = \frac{1}{m} \sum_{i=1}^{m} \phi_j^a \).

Guarantees: Theorem 1

The initial vector \( x^0 \), which is the output of Alg. 1, is a small constant distance \( \delta_b \) away from the true signal \( x^* \), i.e.,

\[
\text{dist}(x^0, x^*) \leq \delta_b \|x^*\|_2,
\]

where \( 0 < \delta_b < 1 \), as long as the number of measurements satisfy \( m \geq C_0 \log mn \) with probability greater than \( 1 - \frac{1}{m} \).

Block CoPRAM - Descent

Setup: Use the initialization \( x^0 \) from Alg. 1.

Objective: Gradually descend to k-block sparse solution \( x^* \).

Challenge: Performance guarantees for convergence.

Solution: (Alg. 2)

Use alternating minimization with model CoSAMP to solve the bi-convex problem:

Phase estimation: \( P^j = \text{proj}(\text{sign}(Ax^j)) \).

Signal estimation: \( x^j \approx \min_{x \in M_k} \|Ax - Py\|_2 \) via model CoSAMP.

Guarantees: Theorem 2

Given an initialization \( x^0 \) satisfying Alg. 1, if number of measurements \( m \geq C_0 \log n \), then the iterates of Alg. 2 satisfy:

\[
\text{dist}(x^{j+1}, x^*) \leq \rho \text{dist}(x^j, x^*),
\]

where \( 0 < \rho_0 < 1 \) is a constant, with probability greater than \( 1 - e^{-\gamma m} \), for positive constant \( \gamma \).

Results

Phase transition: Block CoPRAM v/s CoPRAM, SPARTA, Thresholded Wirtinger Flow (ThWF). CoPRAM is a special case of Block CoPRAM with \( b = 1 \).

Phase transitions for signal length \( n = 3000 \), sparsity \( s = 30 \) and block length \( b = 5 \).

Acknowledgments

This work was supported in part by grants from the National Science Foundation and NVIDIA.

References