Deep One-Bit Compressive Autoencoder

Shahin Khobahi
Arindam Bose
Mojtaba Soltanalian

UIC UNIVERSITY OF ILLINOIS AT CHICAGO
UIC WaveOPT Lab

IEEE SSP 2021
Rio de Janeiro
Brazil

Table of contents

One-Bit Compressive Sensing

- How Compressive Sensing (CS) interacts with a one-bit quantizer?

One-Bit Compressive Sensing

- How Compressive Sensing (CS) interacts with a one-bit quantizer?
- Why?
  - The analysis shifts the focus to bits instead of measurements.
  - One-bit quantizer is an extremely simple and fast device.
  - Fast quantizer allows the compressive acquisition system to take many more measurements.
  - The reconstruction algorithms and the theory are very useful in recovering signals from non-linearly distorted measurements.

Problem Formulation

- One-bit CS acquires a sparse signal $\boldsymbol{x}$ using

$$\boldsymbol{r} = \operatorname{sign}\left(\boldsymbol{\Phi}\boldsymbol{x}\right)$$

Problem Formulation

- One-bit CS acquires a sparse signal $\boldsymbol{x}$ using

$$\boldsymbol{r} = \text{sign}\left(\boldsymbol{\Phi x}\right)$$

- Non-convex problem:

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_0 \quad \text{s.t.} \quad \boldsymbol{r} = \text{sign}\left(\boldsymbol{\Phi x}\right)$$

Problem Formulation

- One-bit CS acquires a sparse signal $\boldsymbol{x}$ using

$$\boldsymbol{r} = \mathrm{sign}\left(\boldsymbol{\Phi}\boldsymbol{x}\right)$$

- Non-convex problem:

$$\begin{aligned} &\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_0 \\ &\text{s.t.} \quad \boldsymbol{r} = \mathrm{sign}\left(\boldsymbol{\Phi}\boldsymbol{x}\right) \end{aligned}$$

- Consistency Principle: $\boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{x} \succeq \boldsymbol{0} \qquad [\boldsymbol{R} = \mathrm{Diag}\left(\boldsymbol{r}\right)]$

Problem Formulation

- One-bit CS acquires a sparse signal $\boldsymbol{x}$ using

$$\boldsymbol{r} = \operatorname{sign}(\boldsymbol{\Phi x})$$

- Non-convex problem:

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_0 \quad \text{s.t.} \quad \boldsymbol{r} = \operatorname{sign}(\boldsymbol{\Phi x})$$

- Consistency Principle: $\boldsymbol{R\Phi x} \succeq \mathbf{0} \qquad [\boldsymbol{R} = \operatorname{Diag}(\boldsymbol{r})]$
- Non-convex ℓ_1-minimization problem on a unit sphere

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1, \quad \text{s.t.} \quad \boldsymbol{R\Phi x} \succeq \mathbf{0}, \quad \|\boldsymbol{x}\|_2 = 1$$

Prior Works

Regularized problem:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 + \alpha\mathcal{R}(\boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{x}) \quad \text{s.t.} \quad \|\boldsymbol{x}\|_2 = 1.$$

Prior Works

Regularized problem:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 + \alpha \mathcal{R}(\boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{x})$$
$$\text{s.t.} \quad \|\boldsymbol{x}\|_2 = 1.$$

Relevant algorithms:

- Renormalized Fixed Point Iteration (RFPI) uses a convex barrier function as a regularizer for the consistency principle.
- Restricted Step Shrinkage (RSS) utilizes a nonlinear barrier function as the regularizer.
- Binary Iterative Hard Thresholding (BIHT) introduces a penalty-based robust recovery algorithm.

The RFPI algorithm

Optimization steps:

$$\boldsymbol{d}_i = \nabla_{\boldsymbol{x}} \mathcal{R}(\boldsymbol{y})\Big|_{\boldsymbol{x}=\boldsymbol{x}_{i-1}} = -\left(\boldsymbol{R}\boldsymbol{\Phi}\right)^T \rho\left(\boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{x}_{i-1}\right),$$

$$\boldsymbol{t}_i = \left(1 + \delta \boldsymbol{d}_i^T \boldsymbol{x}_{i-1}\right) \boldsymbol{x}_{i-1} - \delta \boldsymbol{d}_i,$$

$$\boldsymbol{v}_i = \mathrm{sign}\left(\boldsymbol{t}_i\right) \odot \rho\left(|\boldsymbol{t}_i| - (\delta/\alpha)\mathbf{1}\right),$$

$$\boldsymbol{x}_i = \frac{\boldsymbol{v}_i}{\|\boldsymbol{v}_i\|_2}.$$

where, $\rho(\boldsymbol{y}) \triangleq \mathrm{ReLU}(-\boldsymbol{y}) = \max\{-\boldsymbol{y}, 0\}$.

The RFPI algorithm

Optimization steps:

$$\boldsymbol{d}_i = \nabla_{\boldsymbol{x}} \mathcal{R}(\boldsymbol{y})\Big|_{\boldsymbol{x}=\boldsymbol{x}_{i-1}} = -\left(\boldsymbol{R\Phi}\right)^T \rho\left(\boldsymbol{R\Phi x}_{i-1}\right),$$

$$\boldsymbol{t}_i = \left(1 + \delta \boldsymbol{d}_i^T \boldsymbol{x}_{i-1}\right) \boldsymbol{x}_{i-1} - \delta \boldsymbol{d}_i,$$

$$\boldsymbol{v}_i = \mathrm{sign}\left(\boldsymbol{t}_i\right) \odot \rho\left(|\boldsymbol{t}_i| - (\delta/\alpha)\mathbf{1}\right),$$

$$\boldsymbol{x}_i = \frac{\boldsymbol{v}_i}{\|\boldsymbol{v}_i\|_2}.$$

where, $\rho(\boldsymbol{y}) \triangleq \mathrm{ReLU}(-\boldsymbol{y}) = \max\{-\boldsymbol{y}, 0\}$.

Limitations:

- initial point $\boldsymbol{x}_0$
- step-size δ
- shrinkage threshold $\tau = \delta/\alpha$

The RFPI algorithm

Optimization steps:

$$\begin{aligned} \boldsymbol{d}_i &= \nabla_{\boldsymbol{x}} \mathcal{R}(\boldsymbol{y})\Big|_{\boldsymbol{x}=\boldsymbol{x}_{i-1}} = -\left(\boldsymbol{R}\boldsymbol{\Phi}\right)^T \rho\left(\boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{x}_{i-1}\right), \\ \boldsymbol{t}_i &= \left(1 + \delta \boldsymbol{d}_i^T \boldsymbol{x}_{i-1}\right) \boldsymbol{x}_{i-1} - \delta \boldsymbol{d}_i, \\ \boldsymbol{v}_i &= \operatorname{sign}\left(\boldsymbol{t}_i\right) \odot \rho\left(|\boldsymbol{t}_i| - (\delta/\alpha)\mathbf{1}\right), \\ \boldsymbol{x}_i &= \frac{\boldsymbol{v}_i}{\|\boldsymbol{v}_i\|_2}. \end{aligned}$$

where, $\rho(\boldsymbol{y}) \triangleq \mathrm{ReLU}(-\boldsymbol{y}) = \max\{-\boldsymbol{y}, 0\}$.

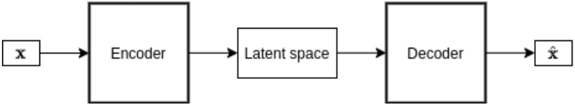
Limitations:

- initial point $\boldsymbol{x}_0$
- step-size δ
- shrinkage threshold $\tau = \delta/\alpha$

Solution

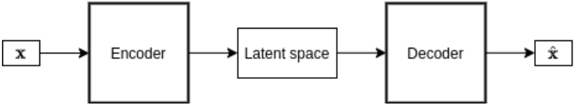
- Deep unfolding can help tuning these parameters by learning from the data.
- We define a decoder function based on the unfolded iterations, and seek to jointly learn the parameters of the proposed autoencoder.

Autoencoder (AE)



- An AE is a generative model comprised of an encoder and a decoder module that are sequentially connected together.

Autoencoder (AE)



- An AE is a generative model comprised of an encoder and a decoder module that are sequentially connected together.
- Encoder: $f_{\Upsilon_1}^{\text{Encoder}} : \mathbb{R}^n \mapsto \mathbb{R}^m$
- Decode: $f_{\Upsilon_2}^{\text{Decoder}} : \mathbb{R}^m \mapsto \mathbb{R}^n$
- $\hat{x} = f_{\Upsilon_2}^{\text{Decoder}} \circ f_{\Upsilon_1}^{\text{Encoder}}(x)$

One-bit Compressive Sensing Autoencoder

Objective:

- Design $\mathbf{\Phi}$ that best captures the information of a K-sparse signal $\boldsymbol{x}$.
- Learn the parameters of the iterative algorithm.

One-bit Compressive Sensing Autoencoder

Objective:

- Design $\boldsymbol{\Phi}$ that best captures the information of a K-sparse signal $\boldsymbol{x}$.
- Learn the parameters of the iterative algorithm.

Encoder:

$$f_{\boldsymbol{\Upsilon}_1}^{\text{Encoder}}(\boldsymbol{x}) = \tilde{\text{sign}}(\boldsymbol{\Phi}\boldsymbol{x}) \qquad [\tilde{\text{sign}}(\boldsymbol{x}) = \tanh(c \cdot \boldsymbol{x}), c > 0]$$

One-bit Compressive Sensing Autoencoder

Objective:

- Design $\mathbf{\Phi}$ that best captures the information of a K-sparse signal $\boldsymbol{x}$.
- Learn the parameters of the iterative algorithm.

Encoder:

$$f^{\text{Encoder}}_{\mathbf{\Upsilon}_1}(\boldsymbol{x}) = \tilde{\text{sign}}(\mathbf{\Phi}\boldsymbol{x}) \qquad [\tilde{\text{sign}}(\boldsymbol{x}) = \tanh(c \cdot \boldsymbol{x}), c > 0]$$

Decoder: Define, $g_{\phi_i} : \mathbb{R}^m \mapsto \mathbb{R}^n$, $\phi_i = \{\boldsymbol{\tau}_i, \delta_i\}$

$$\begin{aligned} g_{\phi_i}(\boldsymbol{z}; \mathbf{\Phi}, \boldsymbol{R}) &= \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \quad \text{with} \\ \boldsymbol{v} &= \tilde{\text{sign}}\left(\boldsymbol{t}\right) \odot \rho\left(|\boldsymbol{t}| - \boldsymbol{\tau}_i\right), \\ \boldsymbol{t} &= \left(1 + \delta_i \boldsymbol{d}^T \boldsymbol{z}\right) \boldsymbol{z} - \delta_i \boldsymbol{d}, \\ \boldsymbol{d} &= -\left(\boldsymbol{R}\mathbf{\Phi}\right)^T \rho\left(\boldsymbol{R}\mathbf{\Phi}\boldsymbol{z}\right) \end{aligned}$$

$$f^{\text{Decoder}}_{\mathbf{\Upsilon}_2}(\boldsymbol{z}_0) = g_{\phi_{L-1}} \circ g_{\phi_{L-2}} \circ \cdots \circ g_{\phi_1} \circ g_{\phi_0}(\boldsymbol{z}_0; \mathbf{\Phi}, \boldsymbol{R}),$$

Loss Function and Training

Proposed loss function:

$$\mathcal{G}(\boldsymbol{x}; \hat{\boldsymbol{x}}) = \underbrace{\sum_{i=0}^{L-1} w_i ||\boldsymbol{x} - \tilde{g}_i(\boldsymbol{x}_i)||_2^2}_{\text{accumulated MSE loss of all layers}} +$$

$$\underbrace{\lambda \sum_{i=0}^{L-1} \mathrm{ReLU}(-[\boldsymbol{\delta}]_i) + \lambda \sum_{i=0}^{nL-1} \mathrm{ReLU}(-[\boldsymbol{\tau}]_i)}_{\text{regularization term for the step-sizes and shrinkage thresholds}}$$

where $\lambda > 0$, $[\delta]_i = \delta_i$, and $\boldsymbol{\tau} = [\boldsymbol{\tau}_0^T, \dots, \boldsymbol{\tau}_{L-1}^T]$.

Numerical Simulations

Specifications:

- $\boldsymbol{x} \in \mathbb{R}^{128}$, $\|\boldsymbol{x}\|_2 = 1$
- $L = 30$
- $\boldsymbol{\Phi} \in \mathbb{R}^{512 \times 128}$

Numerical Simulations

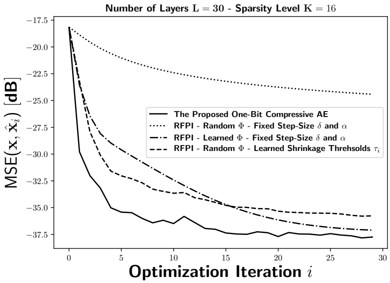
Specifications:

- $\boldsymbol{x} \in \mathbb{R}^{128}$, $\|\boldsymbol{x}\|_2 = 1$
- $L = 30$
- $\boldsymbol{\Phi} \in \mathbb{R}^{512 \times 128}$

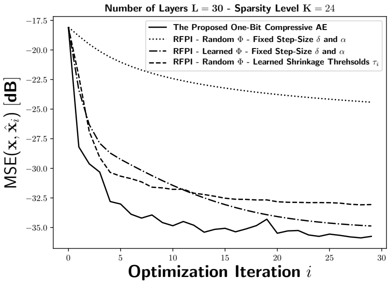
Test Scenarios:

1. *Case 1*: The RFPI algorithm with a randomly generated sensing matrix whose elements are i.i.d and sampled from $\mathcal{N}(0, 1)$, and fixed values for δ, and α.
2. *Case 2:* The RFPI algorithm where the learned $\boldsymbol{\Phi}$ is utilized and the values for δ and α are fixed as the previous case.
3. *Case 3:* The RFPI algorithm with a randomly generated $\boldsymbol{\Phi}$ (same as Case 1), however, the learned shrinkage thresholds vector $\{\tau_i\}_{i=0}^{L-1}$ is utilized with a fixed step size.
4. *Case 4:* The proposed one-bit CS AE method with the learned $\boldsymbol{\Phi}$, $\{\delta_i\}_{i=1}^{L-1}$, and $\{\tau_i\}_{i=0}^{L-1}$.

Numerical Simulations (contd.)



(a)



(b)

Figure: The performance of the proposed method compared to the RFPI algorithm for sparsity levels: (a) $K = 16$, and (b) $K = 24$.

Discussion

- We proposed a hybrid model-based data-driven approach that exploits the existing domain knowledge.
- We unfold state-of-the-art method called RFPI onto the layers of a deep architecture to learn the sensing matrix and also the optimization parameters.
- Our proposed method achieves high accuracy very quickly.

Thank you
and
Questions?

✉: skhoba2@uic.edu