Decentralized joint sparse signal recovery using binary messaging between nodes

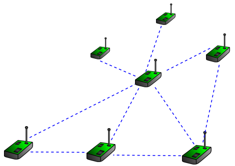
Saurabh Khanna,
Signal Processing for Communication, ECE, IISc

Contents

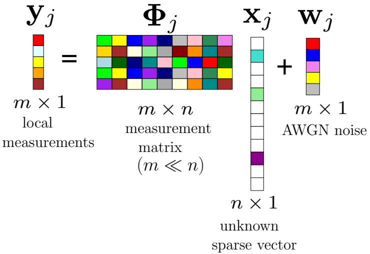
- Problem setup
- Proposed algorithm
- Simulation results
- Future work

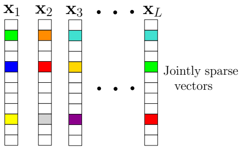
Problem setup

- Network of L sensor nodes
- Single/Multi hop communication links between nodes



- Measurement model at j^{th} node:





- $\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_L$ are **jointly sparse**

- **Goal:**
  - Decentralized estimation of $\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_L$
  - Exploit joint sparsity to reduce no. of local measurements
  - Nodes can exchange only binary vectors

Motivation: why binary messaging?

- Radar sensor fusion for 3D scene reconstruction:
  - # sensors (L) = 4
  - # (range, doppler, angle) hypothesis (N) = 1024 x 32 x 8 = 262144
  - # msg exchanged in each iteration = 12 (fully connected network)
  - # bytes exchanged per iteration = 12 x (262144 x 8) = 24 MB
  - # bytes exchanged per iteration (binary messaging) = 384 KB
- For 802.11g wlan link, typical throughput is 20 Mbps
  - Comm. time per iteration (conventional messaging) = 9.6 seconds
  - Comm. time per iteration (binary messaging) = 0.15 seconds
- Advantages of binary messaging in decentralized algorithms
  - Reduced communication bandwidth requirements
  - Enhanced network lifetime

Past work on joint sparse signal recovery

- Centralized algorithms
  - M-FOCUSS (2005)
  - Distributed Compressed Sensing and SOMP (2005)
  - M-SBL (2007)
- Decentralized algorithms
  - Turbo BCS (2010)
  - MMV-ADM (2011)
  - Decentralized Support detection of MMV with joint Sparsity (Q. Ling and Z. Tian, 2011)
  - Decentralized Bayesian Matching Pursuit (2011)
  - Decentralized Reweighted $\ell_{1/2}$ (2013)
  - DCS-AMP (2013)
  - CB-DSBL (2014)
- Decentralized algorithms with binary messaging
  - Decentralized Subspace Pursuit (2014)
  - Distributed ADMM with 1 bit messaging (GlobalSIP, 2014)

Our work

- A new algorithm called **qCB-DSBL** is proposed for decentralized joint sparse signal recovery which uses binary messaging between nodes
- **qCB-DSBL** stands for Quantized Consensus Based Distributed Sparse Bayesian Learning

Quick recap of SBL

- SBL stands for **Sparse Bayesian Learning** [Wipf and Rao, 2004]
- **Problem:** Recover unknown sparse vector $\mathbf{x}$ from its noisy, underdetermined, linear measurements $\mathbf{y}$

$$\mathbf{y} = \mathbf{\Phi x} + \mathbf{w}$$

  - Impose a sparsity inducing signal prior, $\mathbf{x} \sim \mathcal{N}(0, \mathbf{\Gamma})$
  - $\mathbf{\Gamma} = \mathrm{diag}(\gamma_1, \gamma_2, \dots \gamma_L)$ model the variance of entries of $\mathbf{x}$
  - If $\mathbf{\Gamma}$ is known, from LMMSE theory, $\hat{\mathbf{x}}_{\mathrm{MAP}} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$

$$\mathbf{\Sigma} = \mathbf{\Gamma} - \mathbf{\Gamma \Phi}^T \sigma^2 \mathbf{I}_m + \mathbf{\Phi \Gamma \Phi}^{T\,-1} \mathbf{\Phi \Gamma}$$
$$\boldsymbol{\mu} = \sigma^{-2} \mathbf{\Sigma \Phi}^T \mathbf{y}$$

  - ML estimate $\gamma_{\mathrm{ML}} = \arg\max_{\gamma \in \mathbb{R}_+^n} \log p(\mathbf{y}|\gamma)$ obtained via EM algorithm

$$\text{E step:} \quad Q(\gamma|\gamma^k) = \mathbb{E}_{\mathbf{x}|\mathbf{y},\gamma^k}[\log p(\mathbf{y}, \mathbf{x}|\gamma)]$$
$$\text{M step:} \quad \gamma^{k+1} = \arg\max_{\gamma} Q(\gamma|\gamma^k)$$

Quick recap of CB-DSBL

- CB-DSBL stands for consensus based **Consensus based Distributed Sparse Bayesian Learning**
- MAP estimation of local sparse vectors $\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_L$
- A **common** parameterized Gaussian signal prior $\mathcal{N}(0, \mathbf{\Gamma})$ is assumed by all nodes to induce joint sparsity
- The ML estimate of prior parameters $\mathbf{\Gamma} = \mathrm{diag}(\gamma_1, \gamma_2 \ldots \gamma_n)$ is obtained using EM algorithm
- The M step of EM algorithm is decentralized by using ADMM
- Upon convergence, the nodes arrive at **consensus** with respect to prior parameters $\mathbf{\Gamma}$ resulting in a joint sparse solution

Extending CB-DSBL to use binary messaging

- **Approach-1** Adapt ADMM updates to account for quantized (1 bit) messages
- **Approach-2 qCB-DSBL**
  1. Each node runs SBL iteration to update γ
  2. Each node broadcasts its current estimate of binary support to its ngbd
  3. Each node fuses the binary supports received from its neighboring nodes to generate extrinsic information
  4. Use extrinsic information to update γ
  5. Repeat steps 1 to 4, until convergence

3 questions

1. How to generate local binary support?
2. How to combine binary supports from multiple nodes?
3. How to use extrinsic information to update γ locally at each node?

Q1: How to generate local binary support?

- Assume P_{FA} = Probability of false alarm for zero support detection
  - P_{FA} is applicable on per index basis
  - Same P_{FA} is applicable to all nodes in the network

- At j^{th} node, for index i, $(1 \leq i \leq n)$, we define following two hypothesis

$$\mathcal{H}_0 : \mathbf{x}_j(i) = 0$$

$$\mathcal{H}_1 : \mathbf{x}_j(i) \neq 0$$

or equivalently,

$$\mathcal{H}_0 : \gamma_{\boldsymbol{j}}(i) = 0$$

$$\mathcal{H}_1 : \gamma_{\boldsymbol{j}}(i) > 0$$

where $\gamma_{\boldsymbol{j}}$ denotes the local variance parameters

Q1: How to generate local binary support?

- A log likelihood ratio test (LLRT) is setup as:

  Decide $\mathcal{H}_1$ if

  $$\log \frac{p(\mathbf{y}_j; \mathcal{H}_1)}{p(\mathbf{y}_j; \mathcal{H}_0)} \geq \theta_{j,i}$$

  or equivalently,

  $$(\phi_{j,i}^T(\sigma_j^2 \mathbf{I}_m + \mathbf{\Phi}_j \tilde{\mathbf{\Gamma}}_j \mathbf{\Phi}_j^T)^{-1} \mathbf{y}_j)^2 \geq \theta_{j,i}$$

  where $\tilde{\mathbf{\Gamma}}_j = \sum_{k \neq i} \gamma_{\mathbf{j}}(k) \phi_{j,k} \phi_{j,k}^T$

- Under $\mathcal{H}_0$, $T(\mathbf{y}_j)$ is standard chi-squared distributed (DOF = 1)

  $$T(\mathbf{y}_j) = \frac{(\phi_{j,i}^T(\sigma_j^2 \mathbf{I}_m + \mathbf{\Phi}_j \tilde{\mathbf{\Gamma}}_j \mathbf{\Phi}_j^T)^{-1} \mathbf{y}_j)^2}{\phi_{j,i}^T(\sigma_j^2 \mathbf{I}_m + \mathbf{\Phi}_j \tilde{\mathbf{\Gamma}}_j \mathbf{\Phi}_j^T)^{-1} \phi_{j,i}}$$

  - Denominator in $T(\mathbf{y}_j)$ is a normalization factor
  - Note that $T(\mathbf{y}_j)$ does not depend on $\gamma_j(i)$

Q1: How to generate local binary support?

- Local binary support generated by performing LLRT for all indices $i = 1$ to n:

  Decide $\mathcal{H}_1$ if

$$T(\mathbf{y}_j) = \frac{(\phi_{j,i}^T(\sigma_j^2\mathbf{I}_m + \mathbf{\Phi}_j\tilde{\mathbf{\Gamma}}_j\mathbf{\Phi}_j^T)^{-1}\mathbf{y}_j)^2}{\phi_{j,i}^T(\sigma_j^2\mathbf{I}_m + \mathbf{\Phi}_j\tilde{\mathbf{\Gamma}}_j\mathbf{\Phi}_j^T)^{-1}\phi_{j,i}} \geq \theta_{j,i} = [\mathcal{Q}^{-1}(\frac{P_{FA}}{2})]^2$$

Q2: Combining binary supports from multiple nodes?

- Motivation from cognitive radio literature, how to fuse hard information from multiple sensors
- Goal: Build an optimal (support) detector which fuses hard decisions from multiple sensor(nodes) in a local ngbd
- Possible candidates:
  1. AND rule detector
  2. OR rule detector
  3. K out of N rule detector
- We adopt “K out of N rule” variant i.e., the **majority** rule

Q2: Combining binary supports from multiple nodes?

- Let (Z) denote the "K out of L rule" detector, such that

$$Z = \begin{cases} 0 & \text{if } \frac{L}{2} \text{or more sensor outputs are 0} \\ 1 & \text{if } \frac{L}{2} \text{or more sensor outputs are 1} \end{cases}$$

- Under $\mathcal{H}_0$, sensor outputs are assumed to be $Bernoulli(1 - P_{FA}, P_{FA})$

- Then, $P_{FA}^{Z} = p(Z = 1|\mathcal{H}_0)$ is given by

$$\sum_{l=\frac{L}{2}}^{L} (P_{FA})^{l}(1 - P_{FA})^{L-l}$$

Q3: Local γ update using extrinsic information

- Shrink $\gamma_{\boldsymbol{j}}(i)$ if external binary vector suggests a 0 at i^{th} index

- Question: Shrink $\gamma_{\boldsymbol{j}}(i)$ by how much amount?

- Answer: By shrinking $\gamma_{\boldsymbol{j}}(i)$, we are pursuing a 0 at i^{th} location more aggresively, which will result in reduction of the probability of false alarm for $\mathcal{H}_0$ event.

  So the question is: how much can the local false alarm rate be reduced given the extrinsic support.

- We shrink $\gamma(i)$ (or tighten P_{FA}) such that the resulting P_{FA} equals that of an OR rule detector which fuses the local binary vector and external binary vector

Q3: Local γ update using extrinsic information

- Reduced P_{FA} =
  P_{FA} of OR rule detector (ZZ) which fuses local and external binary vectors

$$
\begin{aligned}
P_{FA}^{ZZ} &= p(ZZ = 1|\mathcal{H}_0) \\
&= p(Z = 1, \text{local decision = 1 for index i}|\mathcal{H}_0) \\
&= p(Z = 1|\mathcal{H}_0)p(\text{local decision = 1 for index i}|\mathcal{H}_0) \\
&= P_{FA}^{Z} P_{FA}
\end{aligned}
$$

- Note that P_{FA}^{ZZ} is tighter than local P_{FA}

- Backpropagating the P_{FA}^{ZZ} to obtain new threshold $\theta_{j,i}^{\text{new}}$

$$\theta_{j,i}^{\text{new}} = [\mathcal{Q}^{-1}(\frac{P_{FA}^{ZZ}}{2})]^2$$

Q3: Local γ update using extrinsic information

- So, for node j and i^{th} index, we have

$$\theta_{j,i}^{\text{old}} = [\mathcal{Q}^{-1}(\frac{P_{FA}}{2})]^2$$
$$\theta_{j,i}^{\text{new}} = [\mathcal{Q}^{-1}(\frac{P_{FA}^{ZZ}}{2})]^2$$

- Define $\eta \triangleq (\frac{\mathcal{Q}^{-1}(0.5P_{FA}^{ZZ})}{\mathcal{Q}^{-1}(0.5P_{FA})})^2 = \frac{\theta_{j,i}^{\text{new}}}{\theta_{j,i}^{\text{old}}}$

- Then, we can write

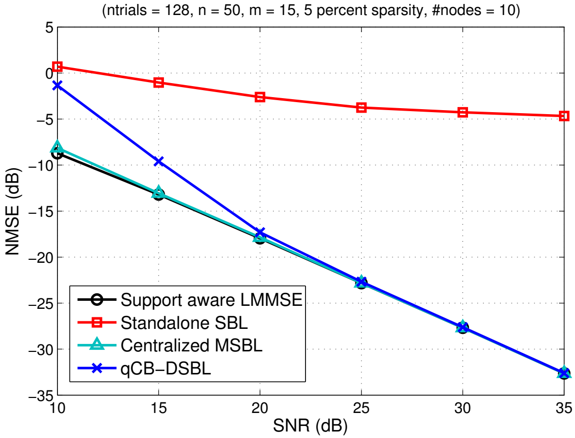
$$\eta = \frac{\theta_{j,i}^{C}(\boldsymbol{\gamma}_{\boldsymbol{j}}^{\text{old}}(k \neq i)) \cdot (\frac{1}{\boldsymbol{\gamma}_{\boldsymbol{j}}^{\text{new}}(i)} + \phi_{j,i}^{T}(\sigma_j^2 \mathbf{I}_m + \boldsymbol{\Phi}_j \tilde{\boldsymbol{\Gamma}}_j^{\text{old}} \boldsymbol{\Phi}_j^{T})^{-1} \phi_{j,i})}{\theta_{j,i}^{C}(\boldsymbol{\gamma}_{\boldsymbol{j}}^{\text{old}}(k \neq i)) \cdot (\frac{1}{\boldsymbol{\gamma}_{\boldsymbol{j}}^{\text{old}}(i)} + \phi_{j,i}^{T}(\sigma_j^2 \mathbf{I}_m + \boldsymbol{\Phi}_j \tilde{\boldsymbol{\Gamma}}_j^{\text{old}} \boldsymbol{\Phi}_j^{T})^{-1} \phi_{j,i})}$$

to get the update rule

$$\boldsymbol{\gamma}_{\boldsymbol{j}}^{\text{new}}(i) = \frac{\boldsymbol{\gamma}_{\boldsymbol{j}}^{\text{old}}(i)}{\eta + (\eta - 1)\boldsymbol{\gamma}_{\boldsymbol{j}}^{\text{old}}(i)(\phi_{j,i}^{T}(\sigma_j^2 \mathbf{I}_m + \boldsymbol{\Phi} \tilde{\boldsymbol{\Gamma}}^{\text{old}} \boldsymbol{\Phi}^{T})^{-1} \phi_{j,i})}$$

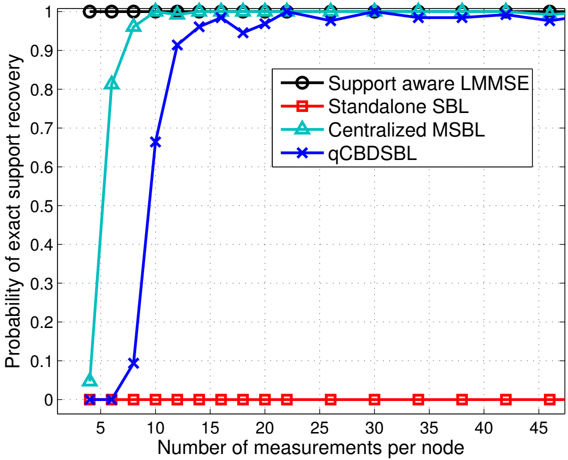
where $\tilde{\boldsymbol{\Gamma}}_j = \sum_{k \neq i} \boldsymbol{\gamma}_{\boldsymbol{j}}(k) \phi_{j,k} \phi_{j,k}^{T}$

MSE performance



- Sim Params: n = 50, m = 15, 10% sparsity, L = 10 nodes, no. of trials = 128, $P_{FA} = 10^{-8}$

Support recovery performance



- Sim Params: n = 50, 10% sparsity, L = 10 nodes, SNR = 20 dB, no. of trials = 128, $P_{FA} = 10^{-8}$

Future work

- How to chose the optimal P_{FA} ?
- Which fusion rule is optimal for generation of extrinsic support ?
- Compare performance with ”DCSP“ and ”DADMM with 1 bit messaging“
- Check performance with more Gaussian sources, unknown noise variance
- Guarantees for convergence/consensus of binary support
- Should we amplify $\gamma_{\boldsymbol{j}}(i)$, if extrinsic information says 1 at i^{th} location
- Derive P_{FA} and P_D for SBL support detector

A forced analogy !