Message Passing Algorithms in Compressed Sensing and Low Density Parity Check Codes

Seyed Mohammad Ebrahim Farhangdoust

Carleton University

November 24, 2014
Introduction to Message Passing Algorithms (MPA)

Applications in Compressed Sensing
Applications in Low Density Parity Check Codes

Problem Statement

$l_p$ norms
The optimization problem

Message Passing Algorithm (SBB)

Analysis of the Algorithm

Proof of Correctness
Running Time Analysis
Basic Idea: Solving an optimization problem with several processors [6]

Algorithms capable of being implemented in parallel processing units [3]

Transferring messages between different nodes of a bipartite graph
Compressed Sensing [1]

- How to compress an $n$–dimensional vector $x$ into an $m$–dimensional vector $y$ where $m << n$ in order to be able to recover $x$ completely.

- This can be done using a matrix transformation.

$$y = Ax$$

where $A \in \mathbb{R}^{m \times n}$
Example (Compressing a Signal)

Let

\[ x = [0, 0, 0.4, 2.1, 0, 0.1]^T \]

\[ A = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0
\end{bmatrix} \]

Then

\[ y = [0.1, 2.5, 0.1, 2.2]^T \]

Question: Is it possible to recover \( x \) from \( y \)?
Let us consider the equivalent linear system of equations:

\[
\begin{align*}
  x_1 + x_5 + x_6 &= 0.1 \\
  x_2 + x_3 + x_4 &= 2.5 \\
  x_1 + x_2 + x_6 &= 0.1 \\
  x_3 + x_4 + x_5 &= 2.2
\end{align*}
\]

- This system has infinite number of solutions.
- Extra knowledge about the signal $x$ is required.
- It has been shown that most of the practical signals are sparse in at least one domain.
Compressed Sensing Recovery Problem

The recovery procedure in Compressed Sensing can be written as:

$$\min_x \#non-zero \text{ elements of } x$$

Subject to $Ax = y$
Coding Theory: Overview

- How to transmit $k$ bits over a telecommunication channel with low probability of error
- Possibility of this transmission: Cloud E. Shannon [5]
- The idea of Coding by Richard W. Hamming [4]
- Adding some extra redundant bits to the original $k$ bits such that there are dependencies between the resulting bits
Example

Example (6,3) Standard Coding

<table>
<thead>
<tr>
<th>Original Data</th>
<th>Transmitted Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000000</td>
</tr>
<tr>
<td>001</td>
<td>001101</td>
</tr>
<tr>
<td>010</td>
<td>010011</td>
</tr>
<tr>
<td>011</td>
<td>011110</td>
</tr>
<tr>
<td>100</td>
<td>100110</td>
</tr>
<tr>
<td>101</td>
<td>101011</td>
</tr>
<tr>
<td>110</td>
<td>110101</td>
</tr>
<tr>
<td>111</td>
<td>111000</td>
</tr>
</tbody>
</table>

The additional bits are chosen such that for every transmitted Data \( c \)

\[ Hc = 0 \]

where

\[
H = \begin{bmatrix}
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]
Some of the bits in the channel may be flipped. Thus:
The received vector in the decoder side:

\[ \hat{c} = c + e \]

Thus,

\[ s = H\hat{c} = Hc + He = He \]

where \( e \) is the unknown noise vector and \( s \) is the syndrome vector.

Again we have an under-determined linear system of equations.

* **Error Correction Capability**
If the decoder finds \( e \) then it can flip the bits in \( \hat{c} \) corresponding to non-zero elements of \( e \).
The most probable error pattern is the sparsest one.
Low Density Parity Check Codes

Decoding Problem

The decoding problem can be written as:

$$\min_x \ #\ non-zero \ elements \ of \ e$$

Subject to \( He = s \)
Norm-$p$ of an $n$–dimensional vector $x$ is defined as:

$$\|x\|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

Some Important norms are:

- norm-2: The Euclidean norm
- norm-$\infty$: Maximum of a vector
- norm-0: $\#$ non-zero elements of a vector
Introduction to Message Passing Algorithms (MPA)

Problem Statement

Message Passing Algorithm (SBB)

Analysis of the Algorithm

$l_p$ norms

The optimization problem

(a) Points with the same norm-2

(b) Points with the same norm-1

(c) Points with the same norm-$\infty$

(d) Points with the same norm-0.5
Compressed Sensing Recovery

The recovery Problem in Compressed Sensing can be rewritten as:

\[
\min_x \|x\|_0 \\
\text{Subject to } Ax = y
\]

Decoding Problem

The decoding problem can be rewritten as:

\[
\min_x \|e\|_0 \\
\text{Subject to } He = s
\]
Example (Compressing a Signal)

Assume that we have the following under-determined linear system of equations

\[ x_1 + 2x_2 = 2 \]

The solutions of this problem lie on the following line:

We want to find the solution with the smallest \( l_0 \) norm. So we start by drawing a small norm-0 in the space and we will blow it up until we reach to an intersection between the line and norm-0.
Introduction to Message Passing Algorithms (MPA)

Problem Statement

Message Passing Algorithm (SBB)

Analysis of the Algorithm

$l_p$ norms

The optimization problem

$x_1 + 2x_2 = 2$
Introduction to Message Passing Algorithms (MPA)
Problem Statement
Message Passing Algorithm (SBB)
Analysis of the Algorithm

1. The optimization problem

\[ x_1 + 2x_2 = 2 \]
Introduction to Message Passing Algorithms (MPA)
Problem Statement
Message Passing Algorithm (SBB)
Analysis of the Algorithm

$l_p$ norms
The optimization problem

$x_1 + 2x_2 = 2$
Problem Statement

Message Passing Algorithm (SBB)

Analysis of the Algorithm

\[ x_1 + 2x_2 = 2 \]

\[ x^* = [0, 1] \]
Preliminaries [7]

- $l_0$ norm minimization is a non-linear optimization problem.
- The $l_0$ minimization algorithms are not practical.
- The proposed MPA will solve the problem in $O(n)$.
- Consider a bipartite graph $G = (V_l \cup V_r, E)$ corresponding to matrix $A$.
- $V_l$, the set of variable nodes, corresponds to the elements of $x$.
  and $V_r$, the set of check nodes, corresponds to the elements of the measured vector $y$.
- For the sake of simplicity we assume that $A$ is sparse and the value of its non-zero elements are chosen randomly from a continuous distribution.
- Also we assume that $x$ is sparse and its non-zero value are chosen randomly from a continuous distribution.
Given the measurement graph $G$ and the values of check nodes, we aim at finding the values of variable nodes in an iterative procedure.

We call a node verified in the algorithm if its value has been determined according to the algorithm.

Once the value of each variable node is verified, this variable node will be removed from the graph and its value will never change during the algorithm.

- Variable node messages: $\mu^v : V_I \mapsto \mathbb{R} \times \{0,1\}$
- Check node messages: $\mu^c : V_r \mapsto \mathbb{R} \times \mathbb{Z}^+$
Algorithm’s Overview [7]

Input: $G = (V_l \cup V_r, E)$, The measured vector $y$
Output: The desired Vector $x$

$V_N \leftarrow \emptyset$
$V_N_{old} = \emptyset$
$\mu^c(i) \leftarrow (y(i), d_c), \ \forall i \in V_r$
$\mu^v(i) \leftarrow (0, 0) \ \forall i \in V_l$

while $V_N \neq V_N_{old}$ or $V_N = \emptyset$ do
   $V_N_{old} \leftarrow V_N$
   Execute Half round 1 Round 1
   Execute Half round 2 Round 1
   Execute Half round 1 Round 2
   Execute Half round 2 Round 2

end while

$x(v) \leftarrow \mu^v_1(v), \ \forall v \in V_l$
for $c \in V_r$ do
    $value \leftarrow y(c) - \sum_{v \in \mathcal{N}(c)} \mu^v_2(v) \mu^v_1(v) A(c, v)$
    $degree \leftarrow d_c - \sum_{v \in \mathcal{N}(c)} \mu^v_2(v)$
    $\mu^c(c) \leftarrow (value, degree)$
end for
for $v \in V_1$ do
  $\text{deg}1 \leftarrow \{c \in \mathcal{N}(v) \mid \mu_2^c(c) = 1\}$
  if $\text{deg}1 \neq \emptyset$ then
    choose $c'$ randomly from $\text{deg}1$
    $\mu^v(v) \leftarrow (\mu_1^c(c')/A(c', v), 1)$
    $VN \leftarrow VN \cup \{v\}$
  else if
    $\exists c_1, c_2 \in \mathcal{N}(v) \mid \mu_1^c(c_1)/A(c_1, v) = \mu_1^c(c_2)/A(c_2, v) \neq 0$ then
    choose $c'$ randomly from one of those check nodes
    $\mu^v(v) \leftarrow (\mu_1^c(c')/A(c', v), 1)$
    $VN \leftarrow VN \cup \{v\}$
  end if
end for
for $c \in V_r$ do

value $\leftarrow y(c) - \sum_{v \in \mathcal{N}(c)} \mu_2^v(v) \mu_1^v(v) A(c, v)$

degree $\leftarrow d_c - \sum_{v \in \mathcal{N}(c)} \mu_2^v(v)$

$\mu^c(c) \leftarrow (\text{value, degree})$

end for
for $v \in V_l$ do
  if $\exists c \in \mathcal{N}(v) \mid \mu_1^c(c) = 0$ then
    $\mu^v(v) = (0, 1)$
    $VN \leftarrow VN \cup \{v\}$
  end if
end for
Example

\[
\begin{array}{cccccccc}
0 & 0.7 & 0.4 & 0.9 & 0.7 & 0.1 & 0 & 0.6 \\
C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 \\
0.4 & 0.5 & 0 & 0.7 & 0.1 & 0 & 0 & 0 \\
\end{array}
\]
Introduction to Message Passing Algorithms (MPA)

Problem Statement

Message Passing Algorithm (SBB)

Analysis of the Algorithm

Example

![Graph Example](link-to-graph)
Example
Example
Example

```
0  0.7  0.4  0.9  0.7  0.1  0  0.6

C1  C2  C3  C4  C5  C6  C7  C8
```

```
V1  V2  V3  V4  V5  V6  V7  V8  V9  V10  V11  V12
0   0   0   0   0   0   0   0   0   0   0   0
```
**Example**

<table>
<thead>
<tr>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_5$</th>
<th>$v_6$</th>
<th>$v_7$</th>
<th>$v_8$</th>
<th>$v_9$</th>
<th>$v_{10}$</th>
<th>$v_{11}$</th>
<th>$v_{12}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
<th>$c_6$</th>
<th>$c_7$</th>
<th>$c_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>0.9</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.6</td>
</tr>
</tbody>
</table>
Example

The image shows a diagram of a message passing algorithm, likely related to a problem in graph theory or network analysis. The nodes are labeled with 'c1' to 'c8' and 'v1' to 'v12', and edges and weights are indicated. The weights on the edges and nodes suggest a computational process, possibly for calculating some form of similarity or connectivity measure.
Example
Introduction to Message Passing Algorithms (MPA)

Problem Statement

Message Passing Algorithm (SBB)

Analysis of the Algorithm

Example

\[ \begin{array}{cccccccc}
0 & 0 & 0.4 & 0.9 & 0 & 0.1 & 0 & 0.6 \\
C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 \\
\end{array} \]

\[ \begin{array}{cccccccc}
\bigcirc_{v_1} & \bigcirc_{v_2} & \bigcirc_{v_3} & \bigcirc_{v_4} & \bigcirc_{v_5} & \bigcirc_{v_6} & \bigcirc_{v_7} & \bigcirc_{v_8} \\
0 & 0.7 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array} \]
Example

\[
\begin{array}{cccccccc}
0 & 0 & 0.4 & 0.5 & 0 & 0.1 & 0 & 0.5 \\
\text{C}_1 & \text{C}_2 & \text{C}_3 & \text{C}_4 & \text{C}_5 & \text{C}_6 & \text{C}_7 & \text{C}_8 \\
\end{array}
\]

\[
\begin{array}{cccccccccccc}
0.4 & 0 & 0.7 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{V}_1 & \text{V}_2 & \text{V}_3 & \text{V}_4 & \text{V}_5 & \text{V}_6 & \text{V}_7 & \text{V}_8 & \text{V}_9 & \text{V}_{10} & \text{V}_{11} & \text{V}_{12} \\
\end{array}
\]
Example

![Graph Diagram]

- Nodes labeled $v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}$
- Edges connecting nodes with weights
- Nodes labeled $C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8$
Example

```
0  0.4  0.5  0  0.1  0  0.5
```

```
C2  C3  C4  C5  C6  C7  C8
```

```
V1  V2  V3  V4  V5  V6  V7  V8  V9  V10  V11  V12
0.4 0.5 0 0.7 0.1 0 0 0 0 0 0 0
```
Lemma (Zero Check Nodes)

*If the value of a check node in each iteration is zero then the values of the variable nodes connected to that check node are all zero with high probability.*

Lemma (Degree 1 Check Nodes)

*If there is at least one degree 1 check node in the neighbourhood of a variable node then the value of the variable node is the value of one of those check nodes.*

Lemma (Equal Check Nodes)

*If there are at least two check nodes in the neighbourhood of a unverified variable node with the same non-zero values then the value of the variable node is the common values of the check nodes with high probability.*
Theorem (False Verification)

The probability of false verification in the algorithm is zero.

Theorem (Algorithm’s Success [2])

If the fraction of non-zero elements in $x$ is less than a certain threshold then when $|V_i| \to \infty$, the algorithm will succeed with probability 1.
Theorem (Algorithm’s Runtime)

The algorithm can be executed in $O(|V_l|)$ time.

Proof.

Every half round of every round consists of a loop which should be distributed over all possible nodes as parallel processors. Therefore, every inner loop will be executed in constant time using parallel processors. Besides, the algorithm will terminate if there is no progress in the number of verified variable nodes. That is the main loop of the algorithm will be executed at most $|V_l|$ times. Therefore, the running time of the algorithm is $O(|V_l|)$.
David L Donoho.
Compressed sensing. 

Yaser Eftekhar, Anoosheh Heidarzadeh, Amir H Banihashemi, 
and Ioannis Lambadaris. 
Density evolution analysis of node-based verification-based 
algorithms in compressed sensing. 

William Gropp, Ewing Lusk, Nathan Doss, and Anthony 
Skjellum. 
A high-performance, portable implementation of the mpi 
message passing interface standard. 
Richard W Hamming.
Error detecting and error correcting codes.

Claude Elwood Shannon.
A mathematical theory of communication.

Joe Whittaker.
*Graphical models in applied multivariate statistics.*

Fan Zhang and Henry D Pfister.
Compressed sensing and linear codes over real numbers.
Thank you