Nonconvex ADMM for Distributed Sparse PCA

Davood Hajienzhad
Joint work Mingyi Hong

Iowa State University

Presented at GlobalSIP 2015
The Main Contribution

**Question:** How to perform principal component analysis over a massively distributed data set?

**Our contribution:** Design and analysis an efficient nonconvex algorithm.
Principal Component Analysis (PCA)

- **PCA** aims to reduce the dimension of multi-variate data set.
- For given data set $D$, the solution of:
  \[
  \max_{x} \| Dx \|_2^2, \quad \text{s.t.} \quad \| x \|_2^2 \leq 1
  \]  
  (1)

  is called first loading vector and the vector $Dx$ is called the first PC [Mackey (2008)] .
- $\| Dx \|_2^2$ represents the explained variance of the first PC.
Sparse PCA

- **Deficiency of PCA:** Most of the PCs’ coefficients are non-zero, making the resulting solutions difficult to interpret.

- **How to address this issue?** Using Sparse PCA (SPCA):

  $$\max_x \|Dx\|_2^2 - \lambda r(x), \quad \text{s.t. } \|x\|_2^2 \leq 1 \quad (2)$$

  where $r(x)$ is a sparsity-promoting, and $\lambda > 0$ controlling the sparsity. [Kwak (2008)].

- $r(x)$ can be: $\|x\|_0$, or its approximations such as $\|x\|_1$ (convex), $\sum_i \log(\epsilon + |x_i|)$ (non-convex).
Literature in SPCA


- [Shen et al (2008)]: Used the connection of PCA with SVD and solved a low rank matrix approximation to extract the PCs (sPCA-rSVD).

- [Journee et al (2010)]: Formulated SPCA as maximization of a convex function on a compact set (G-Power).

Question: Why do we need distributed optimization?

1. Data are collected/stored in a distributed network.
Benefit of Distributed Computing

(2) Memory Limitation
Benefit of Distributed Computing

(3) Privacy Issue
Benefit of Distributed Computing

(4) Parallel Clusters
1. Introduction

2. Distributed SPCA Formulations

3. Proposed ADMM Algorithm

4. Numerical Results
   - Performance on Centralized Data
   - Performance on Distributed Data
Distribution Across the Rows

- Splitting the rows of \( D \in \mathbb{R}^{n \times p} \) into \( N \) sub-matrix:

\[
\text{SPCA problem can be reformulated:}
\]

\[
\max_x \sum_{i=1}^{N} \| D_i x \|_2^2 - \lambda r(x), \quad \text{s.t.} \quad \|x\|_2^2 \leq 1.
\]
Distribution Across the Columns

- Splitting the columns of $D \in \mathbb{R}^{n \times p}$ into $M$ sub-matrix:

$$D = \begin{pmatrix}
A_1 & \cdots & A_{M-1} & A_M
\end{pmatrix}$$

- SPCA problem can be reformulated:

$$\max \left\| \sum_{i=1}^{M} A_i x_i \right\|^2 - \lambda r(x), \quad \text{s.t.} \quad \|x\|_2^2 \leq 1,$$  \hspace{1cm} (4)

- Both formulations are non-convex optimization problem.
| 1 | Introduction |
| 2 | Distributed SPCA Formulations |
| 3 | Proposed ADMM Algorithm |
| 4 | Numerical Results |
|   | - Performance on Centralized Data |
|   | - Performance on Distributed Data |
ADMM setup when rows are distributed

Define new variable $z$:

$$
\begin{align*}
\min_{x,z} & \quad \sum_{i=1}^N -\|D_i x_i\|_2^2 + \lambda r(z) \\
\text{s.t.} & \quad \|z\| \leq 1, \ x_i = z, \ i = 1, \ldots, N;
\end{align*}
$$

(5)

Hong et al. (2014) showed that the ADMM converges to the set of stationary solutions when $r(z)$ is convex.

In our case $r(z)$ is also allowed to be non-convex.
ADMM setup when rows are distributed

- **Augmented Lagrangian function**

\[
L_\rho(x, z; y) = -\sum_{i=1}^{N} \|D_i x_i\|_2^2 + \lambda r(z) + \sum_{i=1}^{N} \langle x_i - z, y_i \rangle \\
+ \sum_{i=1}^{N} \frac{\rho_i}{2} \|x_i - z\|^2
\]

\(y := \{y_i \in \mathbb{R}^p\}_{i=1}^{N}\) is the set of dual variables; \(\rho_i > 0\) is a penalization parameter.

- **ADMM Algorithm**: First, minimizing \(L_\rho(\cdot)\) with respect to \(z\), then with respect to \(\{x_i\}\), followed by an approximate dual ascent update for \(\{y_i\}\) [Boyd et al (2011)].
Non-Convex Regulizer

- **How to deal with non-convex regulizer?** Applying convex approximation technique called the block successive upper-bound minimization (BSUM) [Razaviyayn-Hong-Luo 2013].

- At iteration $t$, regularizer $r(z)$ is replaced with a convex upper-bound approximation, $u(z, v)$ such that:
  1. $u(v, v) = r(v)$
  2. $u'(z, v; d)|_{z=v} = r'(v; d)$
  3. $u(z, v) \geq r(v)$, for all $z, v \in X$.
  4. $u(z, v)$ is continuous $\forall z, v \in X$. 

![Graph](image.png)
For example, upper-bounds for the LSP and M-LSP:

1. The nonconvex LSP, $r(x) = \sum_{j=1}^{p} \log(\epsilon + |x_j|)$.

2. The modified LSP (M-LSP), $r(x) = \log(\epsilon + \|x\|_1)$.

$$u(x, x^t) = \begin{cases} \sum_{j=1}^{p} \frac{1}{\epsilon + |x^t_j|} \left( |x_j| - |x^t_j| \right) & \text{(LSP)} \\ \frac{1}{\epsilon + \|x\|_1} \left( \|x\|_1 - \|x^t\|_1 \right) & \text{(M-LSP)} \end{cases}$$
ADMM algorithm when rows are distributed

Algorithm 1. ADMM for SPCA
Distribute the data into different nodes.
Initialize the variables.
At iteration $t + 1$, do:

S1: The central node updates $z$:

$$z^{t+1} = \arg\min_{\|z\|_2 \leq 1} \lambda u(z, z^t) + \sum_{i=1}^{N} \rho_i/2 \|x_i^t - z + y_i^t/\rho_i\|^2.$$ 

S2: Each node $i$ updates $x_i$ in parallel:

$$x_i^{t+1} = \arg\min_{x_i} \|D_i x_i\|_2^2 + \rho_i/2 \|x_i - z_i^{t+1} + y_i^t/\rho_i\|^2.$$ 

S3: Each node $i$ updates the dual variables in parallel:

$$y_i^{t+1} = y_i^t + \rho_i(x_i^{t+1} - z_i^{t+1}).$$
ADMM setup when columns are distributed

Splitting the columns:

Introducing set of variables \( \{z_i\} \)

\[
\begin{align*}
\min & \quad - \left\| \sum_{i=1}^{M} z_i \right\|^2 + \lambda r(x) \\
\text{s.t.} & \quad \|x\|^2 \leq 1, \quad A_i x_i = z_i, \quad i = 1, 2, \cdots M.
\end{align*}
\]

Augmented Lagrangian:

\[
L_\beta(x, z; y) = - \left\| \sum_{i=1}^{M} z_i \right\|^2 + \lambda r(x) + \sum_{i=1}^{M} \frac{\beta_i}{2} \|A_i x_i - z_i - y_i / \beta_i\|^2.
\]
ADMM algorithm when columns are distributed

Distribute the data $A_i$’s to different nodes.

At iteration $t + 1$

S1: Each node $i$ updates $x_i$ in parallel:

$$\tilde{x}^{t+1}_i = \arg\min_{x_i} \lambda u_i(x_i, x^r_i) + \frac{L_i \beta_i}{2} \|x_i - x^t_i\|^2$$

$$+ \beta_i \langle A_i^T (A_i x^t_i - z^t_i + y^t_i/\beta_i), x_i - x^t_i \rangle$$

S2: Each node sends $c^{t+1}_i = \|\tilde{x}^{t+1}_i\|_2^2$ to the central node.

S3: Central node broadcasts $c^{t+1} = \max\{\sum_{i=1}^M c^{t+1}_i, 1\}$.

S4: Each node computes in parallel: $x^{t+1}_i = \tilde{x}^{t+1}_i / \sqrt{c^{t+1}}$.

S5: The central node updates $z$:

$$z^{t+1} = \arg\min_z - \| \sum_{i=1}^M z_i \|_2^2 + \sum_{i=1}^M \beta_i/2 \|A_i x^{t+1}_i - z_i + y^t_i/\beta_i\|^2.$$ 

S6: Each node $i$ updates the dual variables in parallel:

$$y^{t+1}_i = y^t_i + \beta_i(A_i x^{t+1}_i - z_i^{t+1}).$$
Theorem

We have the following convergence result for Algorithm 1-2:

(1) For Algorithm 1: If $\rho_i \geq 4\|D_i^TD_i\|_2$ for all $i$, then we have:

$$\lim_{t \to \infty} \|x_i^{t+1} - z^{t+1}\| = 0, \; i = 1, \ldots, N.$$  

Further, the algorithm converges to the set of stationary solutions of SPCA.

(2) For Algorithm 2: If $\beta_i \geq 4M$ for all $i$, then we have:

$$\lim_{t \to \infty} \|A_ix_i^{t+1} - z_i^{t+1}\| = 0, \; i = 1, \ldots, M.$$  

Further, the algorithm converges to the set of stationary solutions of SPCA.
1 Introduction

2 Distributed SPCA Formulations

3 Proposed ADMM Algorithm

4 Numerical Results
   - Performance on Centralized Data
   - Performance on Distributed Data
Numerical Results on Pitprops data set

- Centralized version of algorithm \((N = M = 1)\).
- Pitprops data consists of 180 observations and 13 variables.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cardinality</th>
<th>EV</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSPCA [d’Aspremont et al (2007)]</td>
<td>18</td>
<td>79.18</td>
</tr>
<tr>
<td>sPCA-rSVD(\ell_0) [Shen et al (2008)]</td>
<td>18</td>
<td>80.85</td>
</tr>
<tr>
<td>sPCA-rSVD(\ell_1) [Shen et al (2008)]</td>
<td>18</td>
<td>80.40</td>
</tr>
<tr>
<td>Gpower(\ell_0) [Journee et al (2010)]</td>
<td>18</td>
<td>80.64</td>
</tr>
<tr>
<td>Gpower(\ell_1) [Journee et al (2010)]</td>
<td>19</td>
<td>81.11</td>
</tr>
<tr>
<td>BCD-SPCA(\ell_0) [Zhao et al (2015)]</td>
<td>18</td>
<td>80.47</td>
</tr>
<tr>
<td>BCD-SPCA(\ell_1) [Zhao et al (2015)]</td>
<td>18</td>
<td>81.14</td>
</tr>
<tr>
<td>ADMM(\ell_1) [Our Method]</td>
<td>18</td>
<td>82.93</td>
</tr>
<tr>
<td>ADMM(_{MLSP}) [Our Method]</td>
<td>18</td>
<td>83.48</td>
</tr>
</tbody>
</table>
Performance on Distributed Data

Splitting The Rows

- We set \( n = 1,000,000, \ p = 2000 \).

- Randomly generated sparse matrix (95% of elements are zero), a randomly generated dense matrix.

- We split this matrix across the rows into \( N \in \{16, 32, 64\} \) subsets.

- The explained variances in all cases are about 0.064.

<table>
<thead>
<tr>
<th>( N )</th>
<th>Sparse</th>
<th>Dense</th>
<th>Time (Sec)</th>
<th>Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sparse</td>
<td>Dense</td>
<td>Sparse</td>
<td>Dense</td>
</tr>
<tr>
<td>16</td>
<td>1585</td>
<td>1580</td>
<td>40.1</td>
<td>45.3</td>
</tr>
<tr>
<td>32</td>
<td>1574</td>
<td>1574</td>
<td>43.9</td>
<td>117.5</td>
</tr>
<tr>
<td>64</td>
<td>1585</td>
<td>1572</td>
<td>110.1</td>
<td>397.7</td>
</tr>
</tbody>
</table>
**Splitting The Columns**

- Set \( n = 2000 \) and \( p = 100,000 \).
- Let \( M \in \{1, 2, 4, 8, 16, 32, 64\} \).
- Apply Algorithm 2, using the M-LSP regularizer.

### Performance on Distributed Data

<table>
<thead>
<tr>
<th>M</th>
<th>Sparse</th>
<th>Dense</th>
<th>Time (Sec)</th>
<th>Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sparse</td>
<td>Dense</td>
</tr>
<tr>
<td>1</td>
<td>11960</td>
<td>11965</td>
<td>59.90</td>
<td>249.09</td>
</tr>
<tr>
<td>2</td>
<td>11960</td>
<td>11964</td>
<td>43.22</td>
<td>121.19</td>
</tr>
<tr>
<td>4</td>
<td>11962</td>
<td>11965</td>
<td>40.19</td>
<td>80.39</td>
</tr>
<tr>
<td>8</td>
<td>11963</td>
<td>11963</td>
<td>30.58</td>
<td>54.77</td>
</tr>
<tr>
<td>16</td>
<td>11962</td>
<td>11965</td>
<td>23.90</td>
<td>41.61</td>
</tr>
<tr>
<td>32</td>
<td>11962</td>
<td>11964</td>
<td>13.85</td>
<td>25.22</td>
</tr>
<tr>
<td>64</td>
<td>11961</td>
<td>11964</td>
<td>19.75</td>
<td>31.98</td>
</tr>
</tbody>
</table>
Conclusion

- We propose non-convex ADMM algorithms to solve **distributed SPCA** problems.

- Data matrix can be distributed across the **rows** as well as **columns**.

- Our methods deal with **non-convex regularizers** to promote sparsity.
Future Works

- Extend the **star network** to an arbitrary one with non-convex functions.

- Try to find conditions under which we can reach the **global optimal** solution.

- Apply the same way to prove the convergence of ADMM for more non-convex cases.

Thanks for Your Attention.