MATLAB implementation of a scalable spectral clustering algorithm with cosine similarity

Guangliang Chen
San José State University, USA

RRPR 2018, Beijing, China
Introduction

We presented a fast spectral clustering algorithm for large data $X \in \mathbb{R}^{n \times d}$ which is row-sparse ($s \ll d$), or of a moderate dimension ($d \ll n$), e.g.,

- Documents data
- Small images, or data obtained through PCA

when it is appropriate to use the cosine similarity

$$W = XX^T - I \quad \text{(suppose } X \text{ has } L_2\text{-normalized rows})$$
MATLAB Implementation

We showed that one can perform three versions of spectral clustering

Ng, Jordan, and Weiss (NJW), Normalized Cut (NCut), and Diffusion Maps (DM$^{(t)}$)

solely based on very efficient operations on $X$:

- **Elementwise operations**: $O(ns)$ or $O(nd)$ complexity
- **Matrix-vector multiplication**: $O(ns)$ or $O(nd)$ complexity
- **Rank-$k$ SVD**: $O(nsk)$ or $O(ndk)$ complexity

(before the final $k$-means clustering step).

Guangliang Chen | Mathematics & Statistics, San José State University, USA
Algorithm: SSC-cosine (ICPR’18)

Input:

- Data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ (sparse or of moderate dimension, with $L_2$-normalized rows)
- Clustering method (NJW, NCut or DM$^{(t)}$)
- # clusters $k$
- Fraction of outliers $\alpha$

Output: Clusters $C_1, \ldots, C_k$ and a set of outliers $C_0$
Steps:

1. Compute the degree matrix

\[ D = \text{diag}(X(X^T 1) - 1) \]

and remove a fraction \( \alpha \) of points (with lowest degrees) as outliers.

2. Find \( \tilde{X} = D^{-\frac{1}{2}} X \) and its top \( k \) left singular vectors \( \tilde{U} \). Convert \( \tilde{U} \) to \( U = D^{-\frac{1}{2}} \tilde{U} \) (for NCut) or to \( U(t) = D^{-\frac{1}{2}} \tilde{U} \Lambda^t \) (for DM\((t)\)).

3. Normalize the rows of the singular vectors matrix to have unit length and apply \( k \) means to find \( k \) clusters \( C_1, \ldots, C_k \).
Implementation

We implemented the algorithm and conducted all experiments in MATLAB.

Required function: \textit{kmeans} (Statistics and Machine Learning Toolbox) or \textit{litekmeans}$^1$ (by Deng Cai).

Overall time complexity:

- sparse data: $O(nsk)$,
- low-dimensional data: $O(ndk)$.

---

$^1$http://www.cad.zju.edu.cn/home/dengcai/Data/code/litekmeans.m
Small details

- Use sparse matrix operations in MATLAB for sparse input data.
- All multiplications between a matrix and a diagonal matrix (e.g., $D^{-1/2}X$ and $D^{-1/2}\tilde{U}\Lambda^t$) are implemented as element-wise binary operations through the `bsxfun` function in MATLAB.
- The `svds` function is used to find exactly the top $k$ singular values and associated singular vectors of $\tilde{X}$.
- The $k$-means clustering is initialized with the default `plus` option, and uses 10 restarts.
Clustering documents data

20 newsgroups data: 18,774 documents (partitioned into 20 newsgroups), 61,118 words

Table 1: Clustering accuracy (and CPU time in seconds)

<table>
<thead>
<tr>
<th>Projection</th>
<th>Implem.</th>
<th>NJW</th>
<th>NCut</th>
<th>DM(^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>plain</td>
<td>64.0% (137)</td>
<td>64.4% (134)</td>
<td>62.9% (134)</td>
</tr>
<tr>
<td></td>
<td>scalable</td>
<td><strong>65.0% (44.6)</strong></td>
<td>63.8% (43.7)</td>
<td>61.5% (43.7)</td>
</tr>
<tr>
<td>SVD (100)</td>
<td>plain</td>
<td>72.0% (153)</td>
<td>72.1% (161)</td>
<td>73.0% (160)</td>
</tr>
<tr>
<td></td>
<td>scalable</td>
<td>73.6% (15.5)</td>
<td>73.8% (15.8)</td>
<td><strong>74.1% (14.9)</strong></td>
</tr>
</tbody>
</table>

Parameter setting: \(\alpha = 0.01\) (for all algorithms).
## Clustering small images

<table>
<thead>
<tr>
<th>data</th>
<th>size</th>
<th>dim</th>
<th>#cls</th>
<th>plain NJW</th>
<th>scalable NJW</th>
</tr>
</thead>
<tbody>
<tr>
<td>pendigits</td>
<td>10,992</td>
<td>16</td>
<td>10</td>
<td>73.11% (53.7)</td>
<td>73.54% (6.9)</td>
</tr>
<tr>
<td>usps</td>
<td>9,298</td>
<td>256</td>
<td>10</td>
<td>67.44% (29.9)</td>
<td>67.52% (8.0)</td>
</tr>
<tr>
<td>mnist</td>
<td>70,000</td>
<td>784</td>
<td>10</td>
<td>out of memory</td>
<td>52.62% (67.4)</td>
</tr>
</tbody>
</table>
The parameter $\alpha$ (% outliers removed)

Sensitivity study using the 20 newsgroups data:

![Graph showing clustering accuracy vs. fraction of outliers]

- sNJW (SVD 100)
- sNCut (SVD 100)
- sDM1 (SVD 100)
- sNJW (no projection)
- sNCut (no projection)
- sDM1 (no projection)
Thank you for your attention!

**Summary.** We presented the MATLAB implementation of a scalable spectral clustering algorithm with cosine similarity, which consists of only a few basic linear algebra steps. *(Simple, fast, efficient, accurate, and robust)*

**Code and references:**

- GitHub: [https://github.com/glsjsu/rprr2018](https://github.com/glsjsu/rprr2018)
- Paper: Scalable spectral clustering with cosine similarity (ICPR 2018)
- A scalable spectral clustering algorithm based on landmark-embedding and cosine similarity (S+SSPR 2018) ← Gaussian and other similarity

**Contact:** guangliang.chen@sjsu.edu
The math behind the algorithm

We focus on the NJW algorithm for exposition of ideas.

First, we can calculate the degree matrix directly from $A$:

$$D = \text{diag}(W1) = \text{diag}((AA^T - I)1) = \text{diag}(A(A^T1) - 1).$$

Next, we write

$$\tilde{W} = D^{-\frac{1}{2}}(AA^T - I)D^{-\frac{1}{2}} = (D^{-\frac{1}{2}}A)(A^TD^{-\frac{1}{2}}) - D^{-1}
= :\tilde{\Lambda} \tilde{A}^T$$

and remove the low-degree points (which are often outliers!) such that $D^{-1}$ will have an (approximately) constant diagonal.
Consequently, for the remaining data, we use the left singular vectors of \( \tilde{A} \) to approximate the eigenvectors of \( \tilde{W} \).