STREAMING SPECTRAL CLUSTERING WITH KRYLOV BLOCK ITERATION TECHNICAL DOCUMENT

Nguyen Ngoc Khanh, Teh Kah Kuan, Jayakrishnan Melur Madhathil, Sun Hanwu, Tran Huy Dat

1. PROBLEM STATEMENT AND METHOD SUMMARY

The problem is the real-time implementation of spectral clustering algorithm of high dimensional embedding features which achieves top performance for the task of speaker diarisation but is hard to implement in a streaming mode as requiring full matrix operations on Laplacian kernel matrix. This TD proposes a novel iterative way to conduct that operations by applying a trick in cosine similarity kernel decomposition, making it to a tractable form expression to the mathematical problem, to enable the implementation in an iterative streaming fashion and also possible to further adoptions of Krylov low-rank approximation making it even faster and more accurate.

2. SPECTRAL CLUSTERING

Let $X \in \mathbb{R}^{n\times m}$ be the data matrix and a similarity function $k : \mathbb{R}^m \times \mathbb{R}^m \to [0, \infty)$. *Spectral Clustering* constructs the similarity matrix $W \in \mathbb{R}^{n\times n}$ where each entry $W_{ij} = k(x_i, x_j)$. The Laplacian matrix is then calculated as $\mathcal{L} = I - D^{-1/2}WD^{-1/2}$. Based on *Spectral Graph Theory*, the first K eigenvectors of $\mathcal{L}$ well represents data under the similarity function k. *K-means* is then used to form clusters from eigenvectors of $\mathcal{L}$.

3. STREAMING SPECTRAL CLUSTERING WITH BLOCK KRYLOV ITERATION

Figure 1 shows the difference between *Streaming Spectral Clustering* and *Offline Spectral Clustering*.

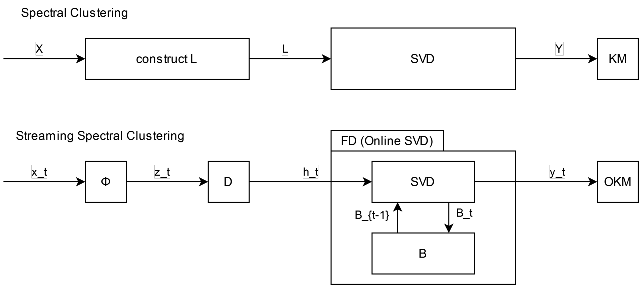

Fig. 1: Speaker Diarization Pipeline

In the online setup, let $X \in \mathbb{R}^{n\times m}$ be the data matrix where each row $x_t \in \mathbb{R}^m, t = 1, 2, ..., n$ is recorded sequentially and a similarity function $k : \mathbb{R}^m \times \mathbb{R}^m \to [0, \infty)$ that is also a kernel. A neat update of the Cholesky decomposition of the Laplacian matrix is presented below.

3.1. Kernel mapping function (ϕ)

Let the similarity function be the cosine function.

$$k(x, y) = \frac{\langle x, y\rangle}{||x||.||y||}$$

The kernel mapping function ϕ transforms each data point $x_t \in \mathbb{R}^m$ into $z_t \in \mathbb{R}^d$ such that $\langle z_i, z_j\rangle = k(x_i, x_j)$. For cosine, the kernel mapping function is

$$z_t = \phi(x_t) = \left(1, \frac{x_t^{(1)}}{||x_t||}, \frac{x_t^{(2)}}{||x_t||}, ..., \frac{x_t^{(m)}}{||x_t||}\right) \in \mathbb{R}^{m+1}$$

3.2. Degree Matrix Approximation (D)

After that, normalizing z_t according to the approximated degree matrix as

$$h_t = D(z_t) \approx \frac{z_t}{\sqrt{\langle z_t, c_t\rangle}}$$

where $c_t = (\sum_{j=1}^{t} z_j)/|| \sum_{j=1}^{t} z_j||$. That yields rows of a matrix $H = (h_1, h_2, ..., h_n)$. The left singular vectors of H coincide with the eigen vectors of the normalized Laplacian $\mathcal{L}$, which enables us to calculate eigenvectors of $\mathcal{L}$ in an online manner by calculating singular vectors of H sequentially.

3.3. Frequent Direction with Block Krylov Iteration (FD_BKI)

FD_BKI (Algorithm 2) is an extension of FD that is used to extract left singular vectors of H sequentially. FD maintains a compression of the stream of data at time $t - 1$ by a sketch matrix $B_{t-1} \in \mathbb{R}^{l\times d}$. When a new data point h_t arrives, SVD algorithm is then used to extract the embedding of the h_t sequentially. FD_BKI adds an additional step (4-6) that both denoises and compresses the input h_t based on the theory of Krylov subspace and Chebyshev polynomials.

3.4. Online K-means (OKM)

Similar to *Offline Spectral Clustering*, *Online K-means* is used to form clusters from singular vectors of H in an online manner.

4. EXPERIMENT

In this section, we describe our implementation of a real-time speaker diarization based on the proposed method.

Algorithm 1 BKI [?]

```
Require: $A \in \mathbb{R}^{n \times d}$, error $\epsilon > 0$, sketch size $l \le n, d$
Ensure: $Z \in \mathbb{R}^{n \times l}$, $P \in \mathbb{R}^{l \times d}$
1: $q \leftarrow \Theta(\frac{\log d}{\sqrt{\epsilon}})$, $\Pi \sim \mathcal{N}(0,1)^{d \times l}$
2: $K \leftarrow [A\Pi, (AA^T)A\Pi, ..., (AA^T)^{q-1}A\Pi] \in \mathbb{R}^{n \times lq}$
3: Orthonormalize columns of $K$ to obtain $Q \in \mathbb{R}^{n \times lq}$
4: Set $U_l \in \mathbb{R}^{lq \times l}$ to be the top-$l$ left singular vectors of $Q^T A$
5: $Z \leftarrow QU_l$, $P \leftarrow Z^T A$
```

Algorithm 2 FD [?] and FD_BKI

```
Require: $H = \{P_t \in \mathbb{R}^{b \times d}\}$, error $\epsilon > 0$, sketch size $l \ll d$, batch size $b$, output dimension $k \le l$
Ensure: $Y = \{U_t \in \mathbb{R}^{b \times k}\}$, $V_t$
1: $B \leftarrow$ empty matrix $\in \mathbb{R}^{0 \times d}$
2: while stream $H$ is not empty do
3:     Read $b$ rows from $H$ to obtain $P_t \in \mathbb{R}^{b \times d}$
4:     if BKI is specified to use and $b > l$ then
5:         $P_t \leftarrow$ BKI$(P_t, \epsilon, l)$
6:     end if
7:     Append $P_t$ into $B$
8:     $U, \Sigma, V \leftarrow$ SVD$(B)$
9:     $V_t \leftarrow V_{[k]}, U_t \leftarrow P_t V_{[k]} \Sigma_{[k]}^{-1}$
10:    if number of rows of $B \ge 2l$ then
11:        $\bar{\Sigma} \leftarrow \text{diag}\left(\sqrt{\max\{0, \sigma^2 - \sigma_l^2\}}\right)$
12:        $B \leftarrow \bar{\Sigma}_{[l]} V_{[l]}^T$
13:    end if
14: end while
```

4.1. Implementation

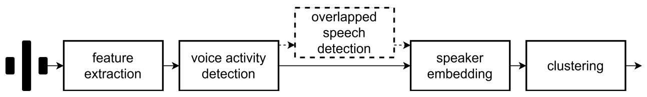

Fig. 2: Speaker Diarization Pipeline

Our speaker diarization pipeline, as illustrated in Figure 2, consists of a feature extraction module converting audio waveform into spectrogram for each frame of 25ms length and 10ms shift. Then voice activity detection (VAD) is applied to isolate the non-speech segments. In this experiment, we adopt Silero VAD [?], a large-scale model-based VAD engine available for public use. Next, the speech segments are passed through a universal background speaker ID model to get its 192-dimensional embedding feature representations, each for 0.96s segments, without overlap. We trained our own background model using ECAPA-TDNN architecture [?] using our own data consisting of more than 6000 thousand speakers, each having around 30 minutes of speech data. Finally, the speech sequence of embedding features goes through a clustering module to get the speaker indices. There was an overlapping speech detection module which is trained on an augmented speaker ID data, to provide extra overlapping detection timing, but it was turned off as we focus on comparing clustering methods.

4.2. Reference Methods

We test our proposed algorithm, which uses Frequent Direction with Block Krylov Iteration (SSC BKI) to construct the low-dimensional embedding of the stream. Three other algorithms are chosen for comparison: offline Spectral Clustering (SC) [?], Streaming Spectral Clustering (SSC) [?] and Online K-means (OKM) [?].

4.3. Experiment Setup

We conducted our experiments on files picked from two free public datasets: (1) IMDA3 [?] and (2) AMI Meeting Corpus [?], all summarized in Table 1.

Table 1: Statistics of the experimental datasets

Dataset	#instances	#features	#clusters
IMDA3	18	192	2
AMI test	24	192	3, 4

To show the stability of algorithms, we performed each 40 times. As the number of speakers are much smaller than the embedding dimension, the number of singular vectors k can be set empirically, according to practical situations. For example, in our experiment, the rooms are small hence we set $k = 8$. The actual number of speakers is then determined by the later online *K-means* step. The size of sketch matrix is set to be $l = 14$ to contain enough information. The number of cosets in online *K-means* is set to be $k \log n$ where n is number of data points.

4.4. Results and Performance Comparison

Table 2: DER(%) performance of the compared algorithms on IMDA3 and AMI dataset

Method	IMDA3	AMI test
SC	20.7	24.4
OKM	28.0	33.3
SSC	20.8	26.1
SSC_BKI	**20.7**	**25.2**

Table 2 shows the general performance of the compared algorithms. For each dataset, we report the average *Diarization Error Rate* (DER). We can make the following observations from these results:

1. Online *K-means* performance is far behind the standard offline *Spectral Clustering* (SC) algorithm.
2. With a compression rate of just around 7%, streaming methods (SSC, SSC-BKI) achieved almost similar results to the standard offline version.
3. Compared with re-computation by offline *Spectral Clustering*, it achieves similar accuracy but with much lower computational cost.
4. With the adoption of *Block Krylov Iteration*, SVD calculation is more robust and accurate, resulting in a slight improvement of DERs.
5. *Block Krylov Iteration* provides faster speed in the implementation, but this needs more comprehensive analysis.