A fast incremental spectral clustering algorithm with cosine similarity

Guangliang Chen

Associate Professor of Statistics & Data Science
Hope College, Holland, Michigan, USA
cheng@hope.edu

ICDM 2023 IncrLearn Workshop
Shanghai, China

Introduction

Spectral clustering is a modern, powerful clustering approach. It uses the eigenvectors of a normalized graph Laplacian for embedding the data into a low-dimensional space for easy clustering.

However, it is well known to face two major challenges:

- scalability (speed and memory),
- out of sample extension.

We present a memory and speed efficient spectral clustering algorithm in the setting of *cosine similarity* that only uses the following efficient linear algebra operations:

- elementwise manipulation
- matrix-vector multiplication
- low-rank SVD

A spectral clustering algorithm

There are different formulations of spectral clustering; here we present the version by Ng, Jordan and Weiss (2001).

Input: Data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, # clusters k, scale parameter σ
Output: Clusters $C_1, \ldots, C_k$

1: Construct a pairwise similarities matrix

$$\mathbf{W} = (w_{ij}), \quad w_{ij} = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2)}, \ i \neq j$$

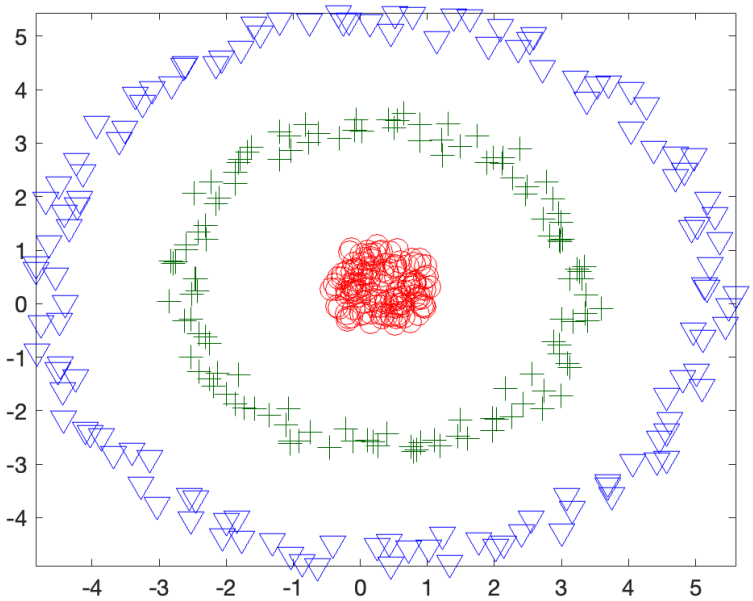
2: Find the row sums of $\mathbf{W}$ and use them to define a diagonal (degree) matrix $\mathbf{D} = \mathrm{diag}(\mathbf{W1})$. Let $\widetilde{\mathbf{W}} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$.

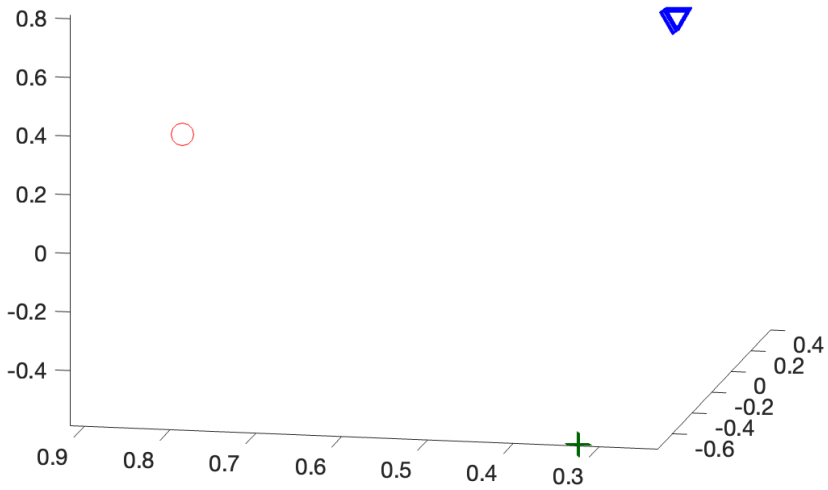
3: Find the k largest eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ of $\widetilde{\mathbf{W}}$ and use them to form an embedding matrix

$$\mathbf{X} \mapsto \quad \mathbf{V} = [\mathbf{v}_1 \ldots \mathbf{v}_k] \in \mathbb{R}^{n \times k}.$$

4: Apply k-means to group the rows of $\mathbf{V}$ (after being normalized to have unit length) into k clusters.

A demonstration





Computational challenges

Spectral clustering is flexible, accurate and powerful, and has been successfully applied to tasks such as image segmentation, documents clustering and social community detection.

However, it is computationally intensive when being applied to large data sets.

- **Memory requirement**: $\mathcal{O}(n^2)$
- **Computational cost**:
  (a) Construction of $\mathbf{W}$: $\mathcal{O}(n^2 d)$
  (b) Decomposition of $\mathbf{W}$: $\mathcal{O}(n^3)$

Consequently, there has been a lot of work on making it scalable in time and/or memory to large data sets.

Speed scalability (ICPR18')

Given (low-dimensional or row-sparse) data $\mathbf{X} \in \mathbb{R}^{n \times d}$ with L_2-normalized rows, the cosine similarity matrix is

$$\mathbf{W} = \mathbf{X}\mathbf{X}^T - \mathbf{I}.$$

First, we can compute the degree matrix $\mathbf{D}$ directly from $\mathbf{X}$ (without needing to construct $\mathbf{W}$):

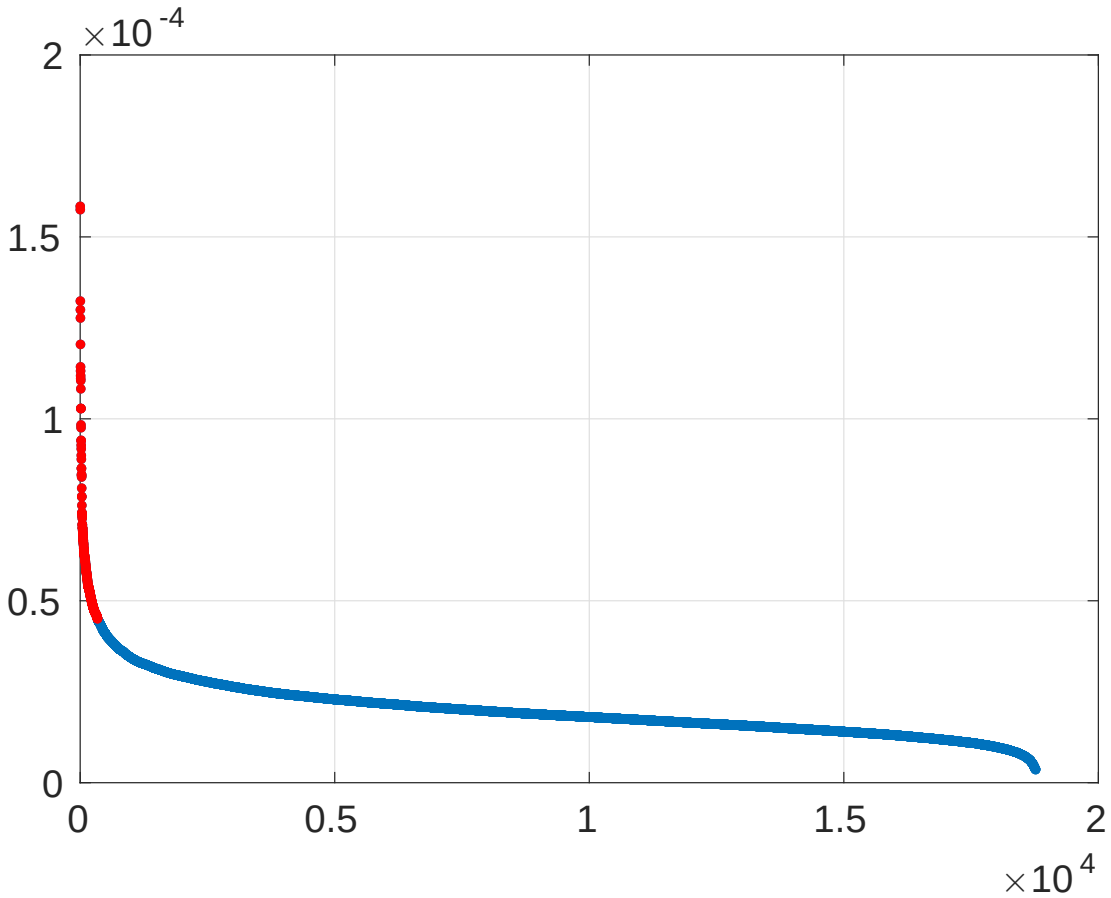
$$\mathbf{D} = \mathrm{diag}(\mathbf{W}\mathbf{1}) = \mathrm{diag}((\mathbf{X}\mathbf{X}^T - \mathbf{I})\mathbf{1}) = \mathrm{diag}(\mathbf{X}(\mathbf{X}^T\mathbf{1}) - \mathbf{1}_n).$$

Next, we write

$$\widetilde{\mathbf{W}} = \mathbf{D}^{-1/2}(\mathbf{X}\mathbf{X}^T - \mathbf{I})\mathbf{D}^{-1/2} = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T - \mathbf{D}^{-1}, \quad \widetilde{\mathbf{X}} = \mathbf{D}^{-1/2}\mathbf{X}.$$

Finally, after removing a small fraction (α) of low-degree points (in order to make $\mathbf{D}^{-1}$ nearly constant diagonal), we use the left singular vectors of $\widetilde{\mathbf{X}}$ to approximate the eigenvectors $\widetilde{\mathbf{U}}$ of $\widetilde{\mathbf{W}}$.

Demonstration: Diagonal entries of $\mathbf{D}^{-1}$ on a large data set (*20 newsgroups*)



The small fraction of points in red have relatively lowest degrees, typically being outliers. They are discarded so that the reduced $\mathbf{D}^{-1}$ matrix is nearly constant diagonal (needed by the approximation).

Memory scalability

Suppose we encounter

- a massive data set, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$, that is too large to be fully loaded into computer memory, but we have access to small batches of the data through sampling, or
- an online data set that arrives sequentially, one point at a time.

In both settings, we would like to perform spectral clustering with the cosine similarity on the whole data set.

We address the memory challenge in the following steps:

1. We first develop a base algorithm using only a **single batch** of data to learn the nonlinear embedding and clustering rule.

Memory scalability

Suppose we encounter

- a massive data set, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$, that is too large to be fully loaded into computer memory, but we have access to small batches of the data through sampling, or
- an online data set that arrives sequentially, one point at a time.

In both settings, we would like to perform spectral clustering with the cosine similarity on the whole data set.

We address the memory challenge in the following steps:

1. We first develop a base algorithm using only a **single batch** of data to learn the nonlinear embedding and clustering rule.
2. We then present an **incremental learning** procedure to continuously refine them as more batches of data are drawn.

Memory scalability

Suppose we encounter

- a massive data set, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$, that is too large to be fully loaded into computer memory, but we have access to small batches of the data through sampling, or
- an online data set that arrives sequentially, one point at a time.

In both settings, we would like to perform spectral clustering with the cosine similarity on the whole data set.

We address the memory challenge in the following steps:

1. We first develop a base algorithm using only a **single batch** of data to learn the nonlinear embedding and clustering rule.
2. We then present an **incremental learning** procedure to continuously refine them as more batches of data are drawn.
3. Lastly, we explain how to **extend** both the nonlinear embedding and clustering rule learned on the training data to the rest of the data in $\mathbf{X}$, as they gradually become available.

Single batch learning

Assume a **small batch** of data of size $s \ll n$, denoted $\mathbf{X}_s \in \mathbb{R}^{s\times d}$, that has become available through sampling. We would like to take advantage of the computing procedure developed in previous work on the *speed scalability* of spectral clustering with cosine similarity.

The following equation establishes the bridge between the full data $\mathbf{X}$ and the sample $\mathbf{X}_s$:

$$\widetilde{\mathbf{X}}^T\widetilde{\mathbf{X}} = \mathbf{X}^T\mathbf{D}^{-1}\mathbf{X} = \sum_{i=1}^{n}\frac{1}{d_i}\mathbf{x}_i\mathbf{x}_i^T \approx \frac{n}{s}\sum_{i=1}^{s}\frac{1}{d_i}\mathbf{x}_i\mathbf{x}_i^T = \frac{n}{s}\widetilde{\mathbf{X}}_s^T\widetilde{\mathbf{X}}_s,$$

where $\widetilde{\mathbf{X}}_s = \mathbf{D}_s^{-1/2}\mathbf{X}_s$ and $\mathbf{D}_s$ represent the restrictions of $\widetilde{\mathbf{X}}$ and $\mathbf{D}$ to the sample $\mathbf{X}_s$, respectively:

$$\mathbf{D}_s = \mathrm{diag}(\mathbf{d}_s), \quad \mathbf{d}_s = \mathbf{X}_s(\mathbf{X}^T\mathbf{1}) - \mathbf{1}_s \approx \frac{n}{s}\mathbf{X}_s(\mathbf{X}_s^T\mathbf{1}_s) - \mathbf{1}_s.$$

The previous equation, $\widetilde{\mathbf{X}}^T\widetilde{\mathbf{X}} = \frac{n}{s}\widetilde{\mathbf{X}}_s^T\widetilde{\mathbf{X}}_s$, allows us to use the rank-k SVD of the sample,

$$\widetilde{\mathbf{X}}_s \approx \widetilde{\mathbf{U}}_s\widetilde{\mathbf{\Sigma}}_s\widetilde{\mathbf{V}}_s^T,$$

to estimate the nonlinear embedding used in the speed-scalable spectral clustering algorithm (ICPR18'), i.e.,

$$Y := \widetilde{\mathbf{U}} = \widetilde{\mathbf{X}}\left(\widetilde{\mathbf{V}}\widetilde{\mathbf{\Sigma}}^{-1}\right) \quad \longleftarrow \quad \widetilde{\mathbf{V}}\widetilde{\mathbf{V}}^T \approx \widetilde{\mathbf{V}}_s\widetilde{\mathbf{V}}_s^T \quad \text{and} \quad \widetilde{\mathbf{\Sigma}} \approx \sqrt{\frac{n}{s}}\widetilde{\mathbf{\Sigma}}_s$$

and apply it to the batch data $\mathbf{X}_s \in \mathbb{R}^{s\times d}$ as follows:

$$\mathbf{Y}_s := \widetilde{\mathbf{X}}_s\left(\widetilde{\mathbf{V}}\widetilde{\mathbf{\Sigma}}^{-1}\right) \approx \widetilde{\mathbf{X}}_s\widetilde{\mathbf{V}}_s\left(\sqrt{\frac{n}{s}}\widetilde{\mathbf{\Sigma}}_s\right)^{-1} = \sqrt{\frac{s}{n}}\widetilde{\mathbf{X}}_s\widetilde{\mathbf{V}}_s\widetilde{\mathbf{\Sigma}}_s^{-1} \in \mathbb{R}^{s\times k}.$$

We then perform k-means clustering in the $\mathbf{Y}_s$ space.

Incremental learning

The embedding obtained on the initial batch of data can be refined by using more samples.

Now, consider a second batch of data $\mathbf{X}_t$ that is independently sampled from the population, and let the combined sample be

$$\mathbf{X}_{s+t} = \begin{bmatrix} \mathbf{X}_s \\ \mathbf{X}_t \end{bmatrix} \in \mathbb{R}^{(s+t)\times d}.$$

By the same reasoning,

$$\widetilde{\mathbf{X}}^T\widetilde{\mathbf{X}} \approx \frac{n}{s+t}\,\widetilde{\mathbf{X}}_{s+t}^T\,\widetilde{\mathbf{X}}_{s+t}.$$

However, we will not compute the SVD of $\widetilde{\mathbf{X}}_{s+t}$ directly as there is a more efficient way of estimate the right singular vectors $\widetilde{\mathbf{V}}$ of $\widetilde{\mathbf{X}}$.

Write

$$\widetilde{\mathbf{X}}_{s+t}^T \widetilde{\mathbf{X}}_{s+t} = \begin{bmatrix} \widetilde{\mathbf{X}}_s^T & \widetilde{\mathbf{X}}_t^T \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{X}}_s \\ \widetilde{\mathbf{X}}_t \end{bmatrix} = \widetilde{\mathbf{X}}_s^T \widetilde{\mathbf{X}}_s + \widetilde{\mathbf{X}}_t^T \widetilde{\mathbf{X}}_t.$$

Using the rank-k SVD of $\widetilde{\mathbf{X}}_s$ (which has already been obtained earlier), we have the following approximation

$$\widetilde{\mathbf{X}}_{s+t}^T \widetilde{\mathbf{X}}_{s+t} \approx \widetilde{\mathbf{V}}_s \widetilde{\mathbf{\Sigma}}_s^2 \widetilde{\mathbf{V}}_s^T + \widetilde{\mathbf{X}}_t^T \widetilde{\mathbf{X}}_t = \underbrace{\begin{bmatrix} \widetilde{\mathbf{V}}_s \widetilde{\mathbf{\Sigma}}_s & \widetilde{\mathbf{X}}_t^T \end{bmatrix}}_{\widetilde{\mathbf{x}}_{k,t}^T} \underbrace{\begin{bmatrix} \widetilde{\mathbf{\Sigma}}_s \widetilde{\mathbf{V}}_s^T \\ \widetilde{\mathbf{X}}_t \end{bmatrix}}_{\widetilde{\mathbf{x}}_{k,t}} = \widetilde{\mathbf{X}}_{k,t}^T \widetilde{\mathbf{X}}_{k,t}$$

We can thus use the rank-k SVD of $\widetilde{\mathbf{X}}_{k,t} \in \mathbb{R}^{(k+t)\times d}$ (which is smaller in size than $\widetilde{\mathbf{X}}_{s+t} \in \mathbb{R}^{(s+t)\times d}$),

$$\widetilde{\mathbf{X}}_{k,t} \approx \widetilde{\mathbf{U}}_{k,t} \widetilde{\mathbf{\Sigma}}_{k,t} \widetilde{\mathbf{V}}_{k,t}^T$$

to estimate $\widetilde{\mathbf{V}}_{s+t}, \widetilde{\mathbf{\Sigma}}_{s+t}$ of $\widetilde{\mathbf{X}}_{s+t}$ and embed $\mathbf{X}_t$ accordingly:

$$\mathbf{Y}_t := \sqrt{\frac{s+t}{n}} \widetilde{\mathbf{X}}_t \widetilde{\mathbf{V}}_{s+t} \widetilde{\mathbf{\Sigma}}_{s+t}^{-1} \approx \sqrt{\frac{s+t}{n}} \widetilde{\mathbf{X}}_t \widetilde{\mathbf{V}}_{k,t} \widetilde{\mathbf{\Sigma}}_{k,t}^{-1}.$$

The stopping criterion

Starting with an initial size s and fixing a step size t, we can iterate the above learning process:

$$s \longleftarrow s + t, \quad \widetilde{\mathbf{V}}_s \longleftarrow \widetilde{\mathbf{V}}_{s+t}$$

Convergence is determined based on Grassmannian metric:

$$g_s = \left\| \widetilde{\mathbf{V}}_{s+t} \widetilde{\mathbf{V}}_{s+t}^T - \widetilde{\mathbf{V}}_s \widetilde{\mathbf{V}}_s^T \right\|_F = \sqrt{2k - 2 \left\| \widetilde{\mathbf{V}}_{s+t}^T \widetilde{\mathbf{V}}_s \right\|_F^2} = \sqrt{2 \sum_{j=1}^{k} \sin^2 \theta_j},$$

where $0 \leq \theta_1 \leq \cdots \leq \theta_k \leq \frac{\pi}{2}$ are the principal angles between the column spaces of $\widetilde{\mathbf{V}}_{s+t}$ and $\widetilde{\mathbf{V}}_s$.

Empirically, we set s such that all $\theta_j \leq \theta_0$, i.e.,

$$g_s < \sqrt{2 \cdot k \cdot \sin^2 \theta_0} = \sqrt{2k} \sin \theta_0.$$

Out of sample extension

Any new point, say $\mathbf{x}_0 \in \mathbb{R}^d$, is embedded as follows:

$$\mathbf{y}_0 = \sqrt{\frac{s}{n}} \left(d_0^{-1/2} \mathbf{x}_0^T \right) \widetilde{\mathbf{V}}_s \widetilde{\mathbf{\Sigma}}_s^{-1} \in \mathbb{R}^k,$$

where

$$d_0 = \mathbf{x}_0^T \sum_{i=1}^{n} \mathbf{x}_i - 1 \approx \frac{n}{s} \mathbf{x}_0^T (\mathbf{X}_s^T \mathbf{1}_s) - 1.$$

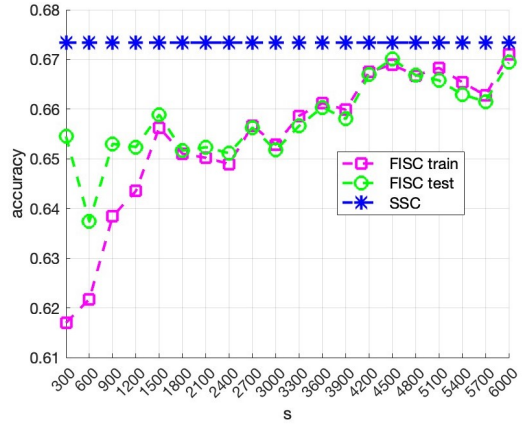
We then assign $\mathbf{y}_0$ to the closest centroids in embedding space.

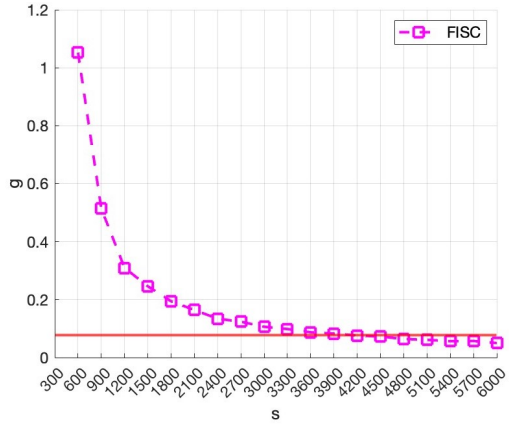
Experimental study

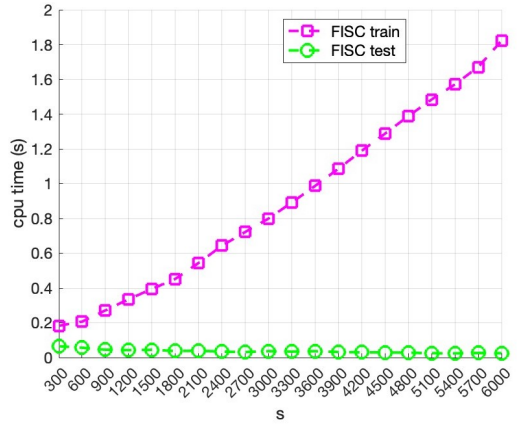
We conduct experiments on the following benchmark datasets to evaluate the performance of our algorithm - Fast, incremental spectral clustering (FISC), in terms of clustering accuracy and CPU time.

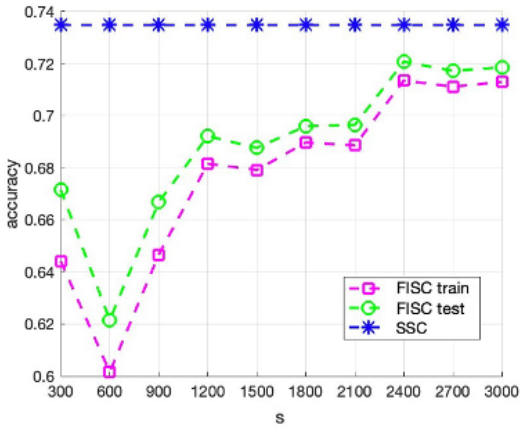
Data sets	n	p	k
usps	9,298	256	10
pendigit	10,992	16	10
mnist	70,000	184	10
20news	18,768	55,570	20
protein	24,387	357	3
covtype	581,012	54	7

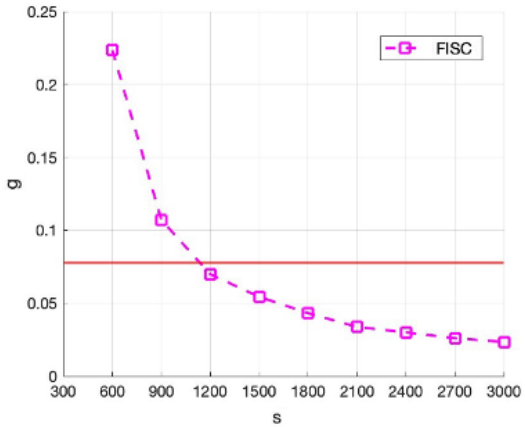
All experiments were conducted with MATLAB R2021b on a desktop computer with 32 GB of RAM and a CPU with 4 cores.

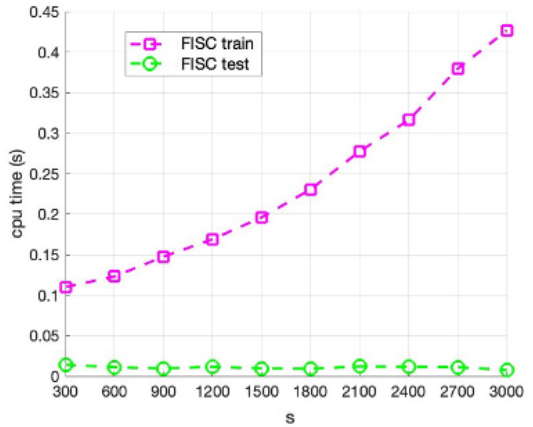

accuracy
s
FISC train
FISC test
SSC

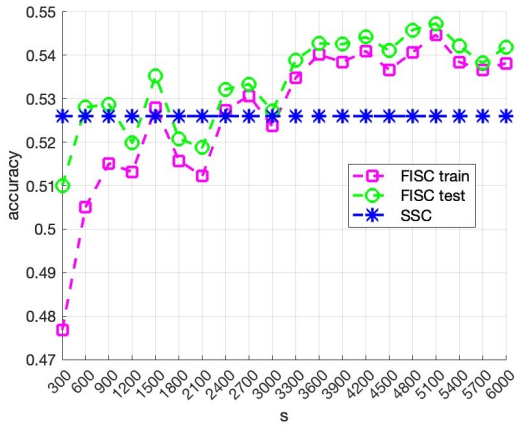

g
s
FISC

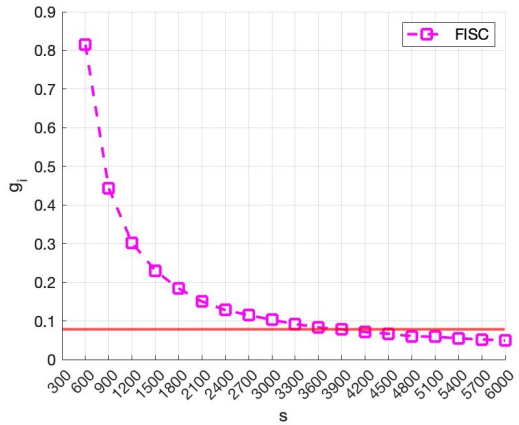

cpu time (s)
s
FISC train
FISC test

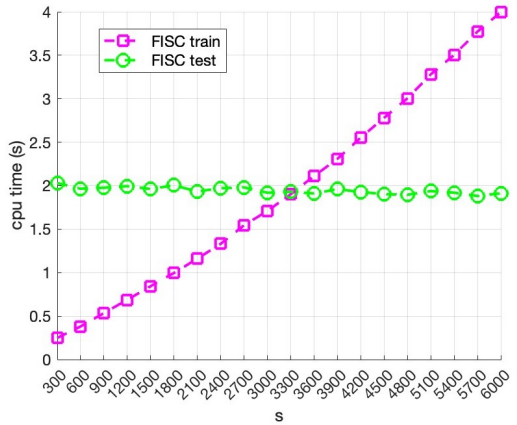

accuracy
s
FISC train
FISC test
SSC

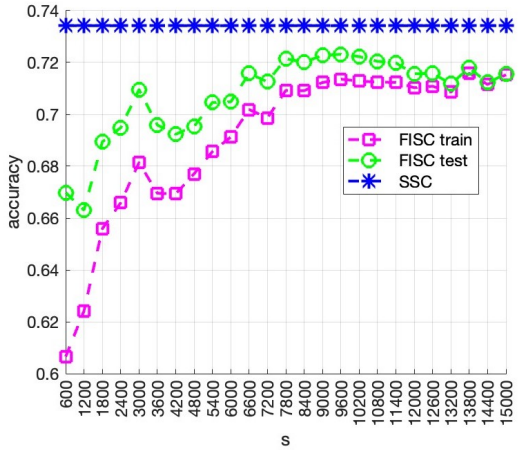

g
s
FISC

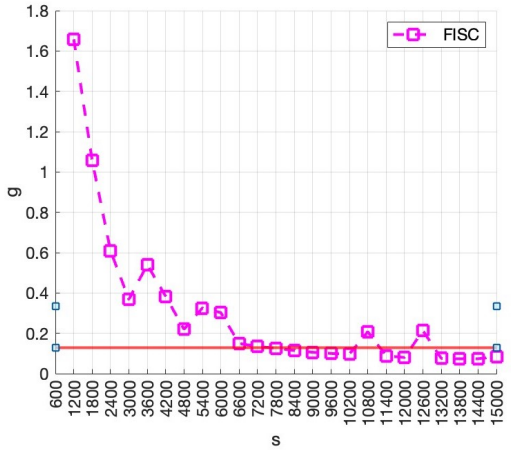

cpu time (s)
s
FISC train
FISC test

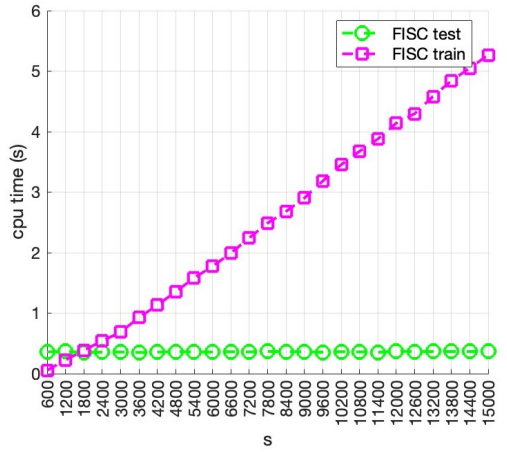

accuracy
s
FISC train
FISC test
SSC

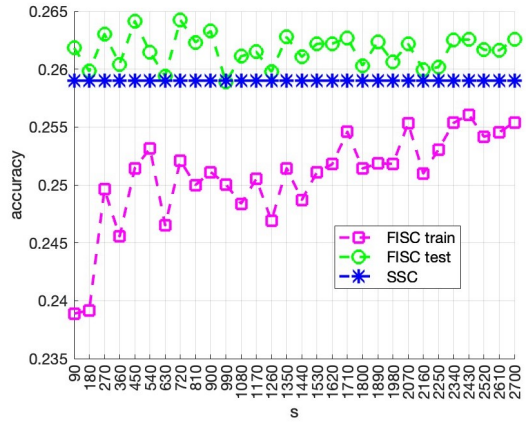

g_i
s
FISC

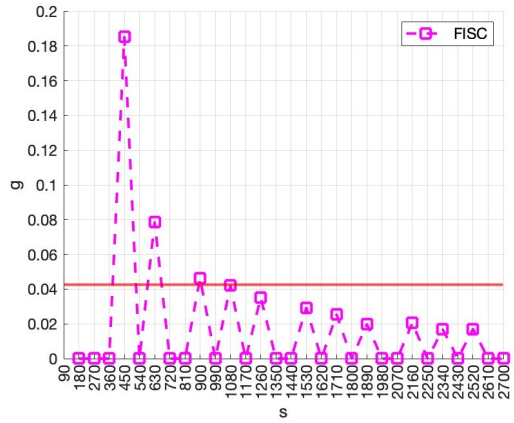

cpu time (s)
s
FISC train
FISC test

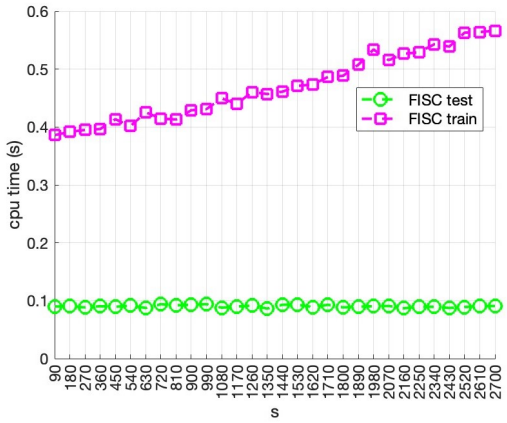

accuracy
s
FISC train
FISC test
SSC

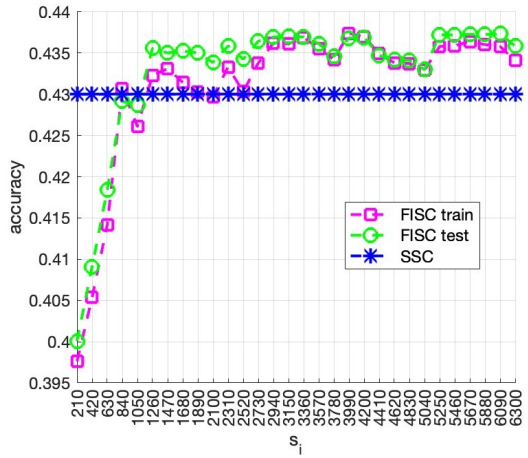

g
s
FISC

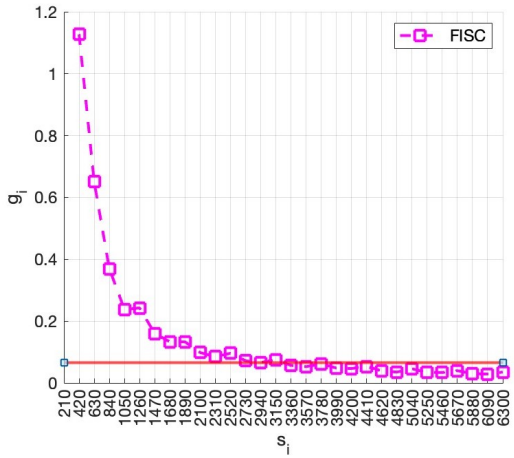

cpu time (s)
s
FISC test
FISC train

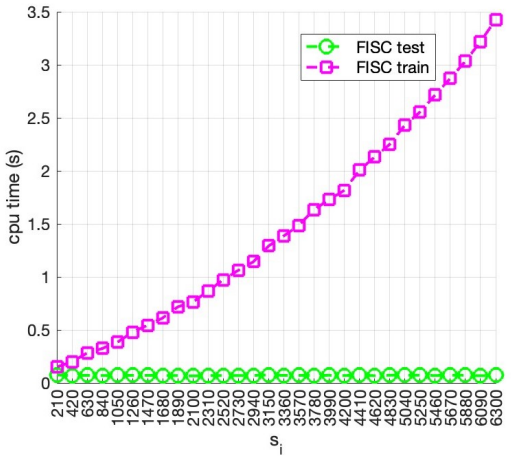

accuracy
s
FISC train
FISC test
SSC


g
s
FISC


cpu time (s)
s
FISC test
FISC train


accuracy
s_i
FISC train
FISC test
SSC


g_i
s_i
FISC


cpu time (s)
s_i
FISC test
FISC train

Thank you for your attention!

Conclusions: We proposed a memory-efficient spectral clustering algorithm that uses small batches of data to effectively learn spectral embedding and clustering maps, as well as the out-of-sample extensions.

In particular, we introduced an incremental procedure that monitors the Grassmannian distances between consecutive iterations to check for convergence.

Future work:

- Statistical analysis of the sampling procedure
- Extension to other kinds of similarity such as Gaussian

Acknowledgement: This was joint work with Ran Li (my former graduate student at San José State University, who now works at Wells Fargo).