Scalable Spectral Clustering via Nyström Approximation: From Synthetic Manifolds to Image Segmentation

Elena Rossi
Politecnico di Torino
Turin, Italy
s349342@studenti.polito.it

Giorgio Zoccatelli
Politecnico di Torino
Turin, Italy
s349395@studenti.polito.it

***Abstract*—This paper analyzes the Spectral Clustering algorithm through theoretical investigation and practical application. We evaluate its ability to identify non-linear manifold structures, comparing the results with the density-based DBSCAN algorithm to highlight qualitative and operational differences. To address the constraints of standard spectral methods on larger datasets, we implement an extension based on the Nyström approximation. This approach is assessed by comparing its clustering fidelity and efficiency against the exact implementation. Finally, the framework is validated through an application to image segmentation, demonstrating its effectiveness in transitioning from synthetic benchmarks to real-world data.**

1. Introduction to Spectral Clustering

A. Context and Mathematical Preliminaries

Clustering is a fundamental task in data science aimed at partitioning vertices or data points into groups based on interaction patterns or similarity. While the traditional k-means algorithm is widely used, it is primarily effective for datasets where clusters are well-separated and near-isotropic, meaning that each cluster has a roughly spherical shape with similar variance in all directions of the feature space. K-means often fails when data possesses complex underlying geometrical structures, such as non-linearly separable concentric circles or half moons, which are some toy examples often exploited when introducing spectral clustering.

Spectral clustering addresses these limitations by taking advantage of the eigenvectors of the graph Laplacian to capture the global and local geometry of the data. It effectively embeds the data into a lower-dimensional space where clusters become more easily separable.

Consider a graph $G = (V, E)$, where $V = \{x_1, \ldots, x_n\}$ is the set of n vertices and E is the set of edges. The similarity between data points is represented by an undirected weighted graph with a adjacency matrix $\mathbf{W} = (w_{ij})_{1 \le i,j \le n}$. Common constructions for the affinity matrix W include:

- ϵ-neighborhood graph: Set $w_{ij} = 1$ if $||x_i - x_j|| \le \epsilon$, and $w_{ij} = 0$ otherwise. This connects only points within a fixed distance, resulting in a sparse graph.
- k-nearest neighbors graph: Connect each point x_i to its k nearest neighbors. Set $w_{ij} = 1$ (or optionally use a similarity kernel) if x_j is among the k nearest neighbors of x_i, and $w_{ij} = 0$ otherwise.
- Fully connected graph: Assign a weight to every pair of points using a Gaussian (RBF) kernel, $w_{ij} = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$ where σ controls the neighborhood width. This results in a dense graph where weights decrease smoothly with distance.

The degree matrix $\mathbf{D}$ is a diagonal matrix where each diagonal entry D_{ii} represents the sum of the weights of the edges connected to vertex i, i.e. $D_{ii} = d_i = \sum_{j=1}^{n} w_{ij}$

The unnormalized graph Laplacian $\mathbf{L}$ is defined as:

$$L = D - W \tag{1.1}$$

The graph Laplacian L exhibits several important algebraic properties:

1) Quadratic Form: For any vector $f \in \mathbb{R}^n$,

$$f^\top L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \tag{1.2}$$

, which shows that L is positive semi-definite.
2) Spectrum: L is symmetric and has n real, non-negative eigenvalues $0 = \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$
3) Connected Components: The multiplicity of the eigenvalue 0 equals the number of connected components of the graph.

To address variations in node degrees, two normalized versions of the Laplacian are commonly employed:

- Symmetric normalized Laplacian: $L_{\text{sym}} = I - D^{-1/2} W D^{-1/2} = D^{-1/2} L D^{-1/2}$.
- Random-walk normalized Laplacian: $L_{\text{rw}} = I - D^{-1} W = D^{-1} L$.

B. The Spectral Clustering Algorithm

The standard procedure for unnormalized spectral clustering involves the following steps:

Algorithm 1 Unnormalized Spectral Clustering

Require: Data points $\{x_i\}_{i=1}^n$, number of clusters k
Ensure: Cluster assignments for each data point

1: Construct the affinity (weight) matrix $W \in \mathbb{R}^{n\times n}$ from the input data.
2: Compute the degree matrix D, where $D_{ii} = \sum_j w_{ij}$.
3: Form the unnormalized graph Laplacian $L = D - W$.
4: Compute the first k eigenvectors $u_1, \ldots, u_k$ corresponding to the k smallest eigenvalues of L.
5: Form the matrix $U \in \mathbb{R}^{n\times k}$ by stacking the eigenvectors as columns.
6: Let $y_i \in \mathbb{R}^k$ denote the i-th row of U.
7: Apply k-means clustering to the points $\{y_i\}_{i=1}^n$.
8: Assign each original point x_i to the cluster corresponding to y_i.

In the practical implementation, we will adopt a refined version of the algorithm, namely the one proposed by Ng, Jordan, and Weiss.

The effectiveness of spectral clustering is rooted in spectral graph theory and, in particular, in the properties of the graph Laplacian. For a connected graph, the smallest eigenvalue $\lambda_1 = 0$ has multiplicity one and its associated eigenspace is spanned by the constant vector $\mathbf{1}$. The second smallest eigenvalue λ_2, known as the *Fiedler value*, is strictly positive and quantifies how well the graph can be partitioned into two weakly connected subgraphs. Its associated eigenvector, the *Fiedler vector*, provides a real-valued embedding of the vertices such that nodes within the same densely connected region tend to have similar values, while nodes across weak connections tend to be separated.

Importantly, even when the graph is connected (and thus has only one zero eigenvalue), the Fiedler vector still carries meaningful clustering information. A small but nonzero value of λ_2 indicates the presence of a nearly disconnected structure in the graph, which can be exploited to identify clusters.

More generally, when the graph consists of k connected components, the Laplacian has exactly k zero eigenvalues and the corresponding eigenspace is spanned by the indicator vectors of these components. In this ideal case, spectral clustering recovers the clusters exactly by selecting the first k eigenvectors.

In practical settings, graphs are typically connected but exhibit approximate cluster structure. Spectral clustering addresses this by embedding the data into the subspace spanned by the first k eigenvectors corresponding to the smallest eigenvalues of the Laplacian. This embedding, often referred to as a *Laplacian eigenmap*, reflects the intrinsic connectivity structure of the graph rather than the original Euclidean geometry of the data.

In this spectral space, clusters that may not be linearly separable in the original feature space often become well separated, allowing simple algorithms such as k-means to recover meaningful partitions.

2. Nyström Method Explanation

As seen in the previous section, the Spectral Clustering algorithm overcomes the intrinsic limits of k-means. However, its application on a large scale remains constrained by high computational costs. In fact, the construction of the affinity matrix $W \in \mathbb{R}^{n\times n}$ and the subsequent spectral decomposition of the Laplacian require a computational complexity of $\mathcal{O}(n^3)$. To overcome this problem, the Nyström method is used.

This method is a technique for finding approximations of eigenvectors and eigenvalues of matrices of large dimensions starting from a subset of samples. The fundamental assumption is that the affinity matrix possesses a reduced rank, meaning that the global relationships between the n points can be reconstructed, without losing much quality, from m representative points called *landmark points*, with $m \ll n$. In this way, the computational complexity is notably reduced, allowing the use of the algorithm even on large real datasets or for high-quality image segmentation.

After selecting a subset of m points from the original dataset, the affinity matrix W is partitioned into blocks:

$$W = \begin{bmatrix} A & B^\top \\ B & C \end{bmatrix} \tag{2.1}$$

with $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{(n-m)\times m}$, and $C \in \mathbb{R}^{(n-m)\times(n-m)}$. In particular, A represents the weight matrix between landmarks, B contains the weights between landmarks and the remaining points, and C contains the weights between the remaining points. Since the case of interest is $n \gg m$, the matrix C is the largest sub-block. Denoting $\hat{U}$ as the approximated eigenvectors of W, the Nyström extension gives:

$$\hat{U} = \begin{bmatrix} U \\ B^T U \Lambda^{-1} \end{bmatrix} \tag{2.2}$$

and the associated approximation of W, which we denote as $\hat{W}$, assumes the form:

$$\hat{W} = \hat{U}\Lambda\hat{U}^T = \begin{bmatrix} A \\ B^T \end{bmatrix} A^{-1} \begin{bmatrix} A & B \end{bmatrix}. \tag{2.3}$$

In this way, the extension implicitly approximates C using $B^T A^{-1} B$.

To ensure computational scalability, the calculation of the degree vector $\hat{\mathbf{d}}$ is performed by exploiting the Nyström block decomposition, avoiding the explicit reconstruction of the full affinity matrix. The successive normalization of the blocks A_{norm} and B_{norm} allows operating directly within the spectral space of the normalized Laplacian.

The first fundamental step consists of determining the degree vector $\hat{\mathbf{d}}$, i.e., the sum of the rows of the approximated affinity matrix $\hat{W}$. By leveraging the block decomposition, the degree vector can be efficiently computed as:

$$\hat{\mathbf{d}} = \begin{bmatrix} \mathbf{a}_r + \mathbf{b}_c \\ \mathbf{b}_r + B^T A^{-1} \mathbf{b}_r \end{bmatrix} \tag{2.4}$$

where $\mathbf{a}_r, \mathbf{b}_r \in \mathbb{R}^m$ are the row sums of A and B respectively, and $\mathbf{b}_c \in \mathbb{R}^{(n-m)}$ is the column sum of B. Once the vector $\hat{\mathbf{d}}$ is obtained, the normalized affinity matrices are derived.

If the matrix A is positive definite, the orthonormal eigenvectors of the complete system are computed through the decomposition of a reduced support matrix Q:

$$Q = A_{norm} + A_{norm}^{-1/2}(B_{norm}^{\top}B_{norm})A_{norm}^{-1/2}. \tag{2.5}$$

Proceeding to the diagonalization of Q, its eigenvectors U and the corresponding diagonal matrix of eigenvalues Λ are obtained:

$$Q = U\Lambda U^{\top}. \tag{2.6}$$

Finally, the eigenvectors of the full normalized system V_{final}, necessary for the clustering phase via k-means, are obtained by projecting the eigenvectors U through the Nyström extension:

$$V_{final} = \begin{bmatrix} A_{norm} \\ B_{norm}^{\top} \end{bmatrix} A_{norm}^{-1/2} U \Lambda^{-1/2}. \tag{2.7}$$

In this way, we obtain orthonormal eigenvectors, providing a robust spectral approximation with a computational cost that is drastically reduced compared to the standard decomposition of the entire Laplacian operator.

The quality of the Spectral Clustering solution and the Nyström approximation depends on the definition of the affinity matrix $\mathbf{W}$. To compute the similarity between two points in the dataset, we have used the Gaussian kernel:

$$\kappa(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2). \tag{2.8}$$

The γ parameter plays a crucial role as it defines the radius of influence of each point. A value of γ that is too high tends to make the points excessively isolated (affinity is nearly zero even for reduced distances), while a low value risks making the system "too connected," blending the borders between natural clusters. To determine the optimal value of γ, different values were tested, evaluating the segmentation quality obtained for each one. For the quantitative validation of the results, the Silhouette Coefficient (S) was employed, defined for each point i as:

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)} \tag{2.9}$$

where a_i represents the average intracluster distance and b_i the minimum average distance to the points of the nearest cluster. The average value of the Silhouette, calculated on a representative sample of the dataset for reasons of computational efficiency, allowed to identify the value of γ that maximizes the internal coherence of the groups and their mutual separation.

Algorithm 2 Nyström-Accelerated Spectral Clustering

Data points $\{x_i\}_{i=1}^n$, number of clusters k, number of landmarks m, parameter γ Cluster assignments for each data point

1: Select a subset of m landmark points from the input data.
2: Construct the affinity sub-matrices $A \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{(n-m)\times m}$.
3: Compute the approximated degree vector $\hat{\mathbf{d}}$ using the Nyström block decomposition.
4: Form the normalized blocks A_{norm} and B_{norm} based on $\hat{\mathbf{d}}$.
5: Construct the reduced support matrix $Q = A_{norm} + A_{norm}^{-1/2}(B_{norm}^{\top}B_{norm})A_{norm}^{-1/2}$.
6: Compute the eigenvectors U and eigenvalues Λ of Q.
7: Compute the full orthonormal eigenvectors $V_{final} = \begin{bmatrix} A_{norm} \\ B_{norm}^{\top} \end{bmatrix} A_{norm}^{-1/2} U \Lambda^{-1/2}$.
8: Let $y_i \in \mathbb{R}^k$ denote the i-th row of the first k columns of V_{final}.
9: Apply k-means clustering to the points $\{y_i\}_{i=1}^n$.
10: Assign each original point x_i to the cluster corresponding to y_i.

3. Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

A. Context and Mathematical Preliminaries

Unlike partitioning methods such as k-means, which are designed to minimize within-cluster variance and effectively assume clusters to be convex and near-isotropic, density-based clustering is built upon the premise that clusters are dense regions of points in the data space separated by regions of lower point density. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) formalizes this intuition by categorizing points based on their local neighborhood density, allowing for the discovery of clusters with arbitrary shapes and the robust identification of outliers.

Consider a database D consisting of n points in a d-dimensional space. The density-based model is parameterized by a radius $\epsilon \in \mathbb{R}^+$ and a density threshold $minPts \in \mathbb{N}$. The fundamental mechanics of the algorithm are defined through the following mathematical constructs:

1) ϵ-neighborhood: The ϵ-neighborhood of a point p, denoted by $N_\epsilon(p)$, encompasses all points in D whose distance from p is less than or equal to ϵ:
$$N_\epsilon(p) = \{q \in D \mid dist(p, q) \leq \epsilon\} \tag{3.1}$$
where $dist(p, q)$ represents a distance metric, typically the Euclidean distance.
2) Core Point: A point p is classified as a *core point* if its ϵ-neighborhood contains a minimum number of points, satisfying $|N_\epsilon(p)| \geq minPts$. Core points are the internal building blocks of a cluster.
3) Direct Density-Reachability: A point p is said to be *directly density-reachable* from a point q if $p \in N_\epsilon(q)$

and q is a core point. This definition implies that p is within the "reach" of a dense region centered at q.

4) Density-Reachability: A point p is *density-reachable* from q if there exists a sequence of points $p_1, \ldots, p_n$ such that $p_1 = q$ and $p_n = p$, where each p_{i+1} is directly density-reachable from p_i. This transitive property allows the algorithm to merge overlapping dense regions.
5) Density-Connectivity: Two points p and q are *density-connected* if there exists a point o such that both p and q are density-reachable from o. Density-connectivity is a symmetric relation that formally defines the extent of a cluster.

Formally, a cluster C is a non-empty subset of the database D that satisfies the requirements of *maximality* (if $p \in C$ and q is density-reachable from p, then $q \in C$) and *connectivity* (all points in C are density-connected to one another). Points that do not satisfy these conditions for any cluster are labeled as noise.

B. The DBSCAN Algorithm

The DBSCAN algorithm operates by exploring the dataset and expanding clusters from core points. When an unclassified point is found to be a core point, a new cluster is initialized, and all points in its ϵ-neighborhood are added to a processing queue. The cluster grows as the algorithm recursively explores the density-reachable points from every core point added to the set.

Algorithm 3 DBSCAN Algorithm

Require: Database D, radius ϵ, threshold $minPts$
Ensure: Cluster labels for points in D

Initialize all points in D as *unclassified*
for each point $p \in D$ **do**
  if p is *unclassified* **then**
    $N \leftarrow$ RANGEQUERY(D, p, ϵ)
    if $|N| < minPts$ **then**
      Mark p as *noise*
    else
      $c \leftarrow$ generate new cluster label
      $p.label \leftarrow c$
      $S \leftarrow N \setminus \{p\}$
      for each $q \in S$ **do**
        if $q.label =$ noise **then** $q.label \leftarrow c$
        end if
        if q is *unclassified* **then**
          $q.label \leftarrow c$
          $N_q \leftarrow$ RANGEQUERY(D, q, ϵ)
          if $|N_q| \geq minPts$ **then** $S \leftarrow S \cup N_q$
          end if
        end if
      end for
    end if
  end if
end for

The computational efficiency of DBSCAN is primarily determined by the execution of the `RANGEQUERY` operation. In a naive implementation, the complexity is $O(n^2)$ due to the pairwise distance calculations required for each point. It is important to note that in higher dimensions ($d \geq 3$), the complexity often degrades, with theoretical lower bounds reaching $\Omega(n^{4/3})$ for Euclidean metrics, indicating that the index performance is highly dependent on the data distribution and the intrinsic dimensionality of the dataset.

C. Parameter Selection and Diagnostic Heuristics

Selecting appropriate values for ϵ and $minPts$ is critical for obtaining meaningful clustering results. A widely adopted heuristic for estimating ϵ involves the sorted k-dist graph. For a fixed k (typically set to $minPts-1$), the distance to the k-th nearest neighbor is computed for every point. These distances are then plotted in descending order. The elbow of the resulting curve indicates a threshold where the density significantly drops, representing the transition from points within clusters to noise points.

Furthermore, several diagnostics can be used to validate the chosen parameters. A clustering result where a single large cluster contains more than 50% of the data may indicate that ϵ is too large, causing distinct clusters to merge. Conversely, a noise percentage significantly higher than 30% suggests that ϵ might be too small or $minPts$ too high, preventing the algorithm from identifying valid dense regions. In environments with varying densities, hierarchical extensions such as OPTICS may be required to complement the basic DBSCAN framework.

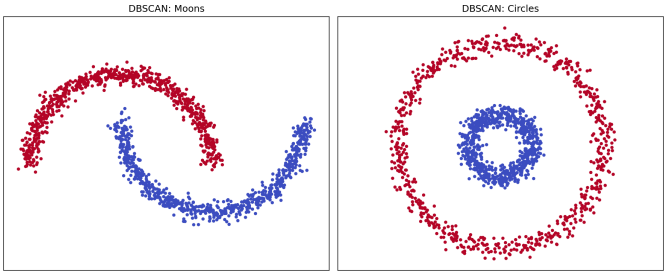
4. Practical Implementation

The practical implementation of the proposed theoretical framework unfolds in two distinct phases: an initial validation on synthetic datasets—specifically 'half moons' and 'concentric circles'—followed by a real-world application focused on image segmentation.

A. Synthetic Benchmarks

In the initial phase involving synthetic benchmarks, we observed that the standard K-means algorithm fails to correctly classify the clusters in both datasets, as anticipated. This limitation arises because K-means relies on Euclidean distance, implicitly assuming spherical and convex cluster shapes. Consequently, it struggles with the highly anisotropic geometry inherent in these examples.

Subsequently, we deployed a custom implementation of the DBSCAN algorithm, developed from scratch in strict accordance with the theoretical framework. In stark contrast to K-means, this density-based approach proved successful, correctly identifying the complex cluster structures.



Building upon these initial results, we now detail the specific design choices adopted for our Spectral Clustering implementation. We stick to the normalized spectral clustering approach proposed by Ng, Jordan, and Weiss (NJW).

The construction of the similarity graph is a critical step that defines the local connectivity of the data. While standard approaches often utilize a fully connected graph with Gaussian weights, our implementation introduces a sparsification step to mitigate noise and reduce computational complexity. We computed the affinity matrix W utilizing a Gaussian Radial Basis Function (RBF) kernel, controlled by the scaling parameter σ. However, rather than retaining all connections, we applied a k-Nearest Neighbors (k-NN) filter. For each data point x_i, we retained edges only to its k nearest neighbors.

To address the inherent asymmetry of the k-NN relationship (where x_i being a neighbor of x_j does not imply the reverse), we enforced symmetry on the adjacency matrix by taking the maximum of the mutual weights:

$$W_{ij}^{sym} = \max(W_{ij}, W_{ji}) \tag{4.1}$$

This strategy ensures that the resulting graph is undirected and that strong local connections are preserved, even if they are unidirectional during the initial neighbor search.

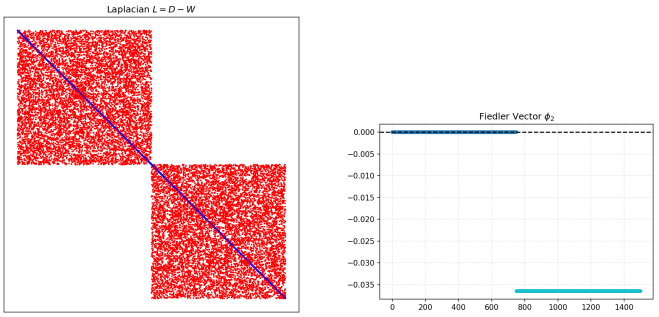
Regarding the graph Laplacian, we selected the symmetric normalized Laplacian, defined as $L_{sym} = I - D^{-1/2}WD^{-1/2}$, due to its superior stability when handling clustering distributions with varying densities.

An important feature of our implementation, strictly coherent with the NJW algorithm, is the row-normalization of the eigenvector matrix. After extracting the first k eigenvectors corresponding to the smallest eigenvalues and stacking them into a matrix $U \in \mathbb{R}^{N\times k}$, we normalized each row to unit length:

$$U_{ij} \leftarrow \frac{U_{ij}}{\sqrt{\sum_{l=1}^{k} U_{il}^2}} \tag{4.2}$$

This step projects the data points from the spectral domain onto the unit hypersphere. This projection is crucial as it compels the clustering algorithm to focus on the angular distribution of the points rather than their magnitude, thereby correcting for variations in node degrees within the clusters. Finally, the clustering assignment was performed by applying the K-means algorithm to these normalized features, effectively partitioning the data in the spectral space.

To verify the effectiveness of our graph construction and spectral embedding, we inspected the intermediate representations generated during the algorithm's execution. Figure in



the left panel above illustrates the sparsity pattern of the graph Laplacian derived from the 'concentric circles' dataset. The matrix exhibits a distinct block-diagonal structure, which confirms that our parameter selection for the RBF kernel and the k-NN sparsification successfully disconnected the two concentric rings. By removing noisy edges between the inner and outer circles, the algorithm effectively isolated the clusters within the graph topology.

This separation is further evidenced by the spectral embedding itself. Figure in the right panel above plots the components of the Fiedler vector (the eigenvector ϕ_2 associated with the second smallest eigenvalue). As predicted by spectral graph theory for disconnected or weakly connected components, the vector approximates a step function. The indices corresponding to the two clusters are mapped to distinct, nearly constant values. Consequently, the data—originally non-linearly separable in Euclidean space—becomes linearly separable in this 1D spectral embedding, rendering the final K-means step trivial.

It is important to note that while the figures presented here specifically depict the 'concentric circles' benchmark, we observed consistent spectral properties and an equally clean separation for the 'half moons' dataset. These additional visualizations, along with further analyses, are detailed in the related computational notebook.

In implementing the Nyström Method to the synthetic datasets, several specific design choices were made to ensure numerical stability and automation of parameter selection. We employed uniform random sampling without replacement to select the m landmark points.

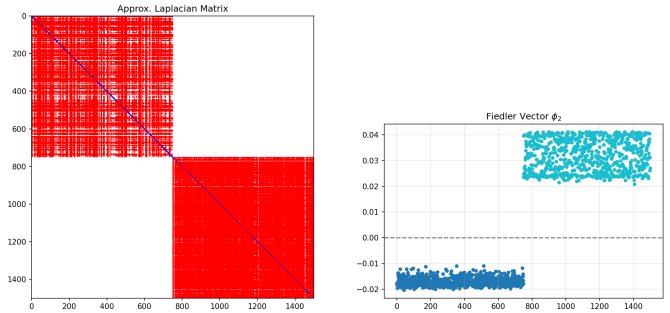
As we defined in 2.8 the performance of the algorithm is highly sensitive to the parameter γ. We implemented a dual-stage selection process:

- Heuristic Baseline: By default, the kernel width is dynamically initialized as $\gamma = [2 \cdot \text{median}(\text{Var}(X))]^{-1}$. This provides a robust starting point that automatically scales with the variance of the data features without requiring manual tuning.
- Silhouette-based Optimization: To refine the clustering quality, we implemented a dedicated routine (`gamma_best`) that evaluates multiple γ candidates. Following the logic of Eq. (2.9), the algorithm selects the value that maximizes the Silhouette Score. To maintain the computational efficiency typical of the Nyström

method, this score is calculated on a representative random subsample of the dataset.

Crucially, in the implementation of the degree vector $\hat{d}$ and the support matrix Q, our implementation utilizes the Moore-Penrose pseudo-inverse—implemented via `np.linalg.pinv`—instead of the standard matrix inverse. This choice significantly enhances stability, preventing the algorithm from failing when the sub-matrix A is singular or near-singular, a common occurrence in sparse or distinctively clustered data. Furthermore, a regularization term $\epsilon = 10^{-10}$ was added during the computation of the inverse square root degrees ($D^{-1/2}$) and a row normalization to avoid division by zero.

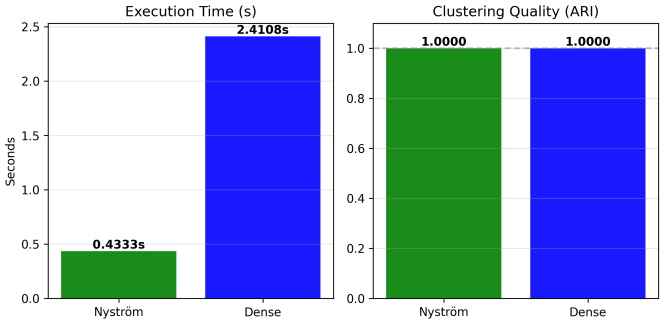
Finally, consistent with the standard Spectral Clustering pipeline, we selected the top k eigenvectors, enforced row-normalization to unit length, and applied K-means to partition the data in the approximated spectral embedding.



To assess the impact of the low-rank approximation on the spectral embedding, we analyzed the intermediate matrix representations generated by the Nyström method.

The Figure in the left panel above displays the approximate Laplacian matrix reconstructed from the landmark points. Similar to the exact Spectral Clustering implementation, the matrix exhibits a clear block-diagonal structure, indicating successful isolation of the concentric rings. However, a key difference is observable in the matrix density. Unlike the strictly sparse graph produced by the k-NN filter in the previous section, the Nyström approximation reflects the dense nature of the Gaussian RBF kernel. While the off-diagonal blocks effectively vanish due to the exponential decay of the kernel (controlled by the dynamic γ), the intra-cluster connections remain fully populated, resulting in the visible dense blocks.

The consequences of the approximation are further visible in the spectral domain. Figure in the right panel plots the approximate Fiedler vector. While the step-function behavior is preserved—clearly distinguishing the two clusters with positive and negative values—the signal is noticeably noisier than in the exact method. We observe significant intra-cluster variance, which serves as a visual proof of the sampling error and the reconstruction process. Despite these oscillations, the spectral gap remains robust. The separation between the two bands of values is sufficiently large that the subsequent K-means algorithm can perfectly partition the data, confirming that the Nyström method retains the discriminative power of the spectral embedding while drastically reducing computational costs.



To complement the qualitative visual analysis, we performed a quantitative comparison between the exact Spectral Clustering implementation ("Dense") and the Nyström approximation. The results, summarized in the Figure above, evaluating both the computational efficiency and the clustering accuracy.

To strictly measure the accuracy, we utilized the *Adjusted Rand Index* (ARI). The ARI is a measure of the similarity between two data clusterings—in this case, the predicted labels versus the ground truth classes. Unlike the standard accuracy metric, which is sensitive to change of label indices, ARI counts pairs of points that are assigned in the same or different clusters in the predicted and true partitionings. It is adjusted for chance, meaning that: a score of 1.0 indicates a perfect match, while a score close to 0.0 indicates random labeling independent of the true classes.

As evidenced by the right panel of Figure above, both the Dense and Nyström methods achieved an ARI of 1.0. This confirms that the noise observed in the approximate Fiedler vector for the Nystrom Method was indeed negligible regarding the final partitioning; the Nyström approximation successfully recovered the exact cluster structure.

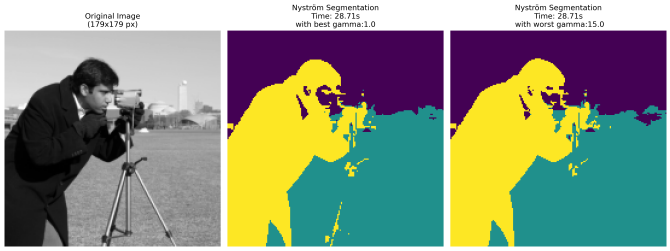
The left panel, however, highlights the critical advantage of the approximation. The Nyström implementation completed the task in 0.433s, compared to 2.411s for the exact method—a speedup factor of approximately $5.5\times$. This result empirically validates the scalability of the proposed framework: by operating on a low-rank approximation of the affinity matrix, we achieved a significant reduction in computational overhead while maintaining the same level of clustering fidelity on non-linear manifolds.

B. Image segmentation

This performance difference becomes critical when performing real Image Segmentation: for the coins dataset, we observed that the Nyström method is much faster than the dense method, while visually maintaining the same quality.

Finally, we implemented a "stress test" using a high-resolution image of the photographer. In this case, the classic method failed due to an out-of-memory error. The necessity to compute an affinity matrix with a size equal to the square of the number of pixels made the standard method infeasible; the Nyström

method, conversely, succeeded in completing the clustering. However, it is crucial to emphasize that the quality of the segmentation is strongly dependent on the choice of the RBF kernel parameter, γ. As previously discussed, this parameter controls the width of the Gaussian kernel and defines the local connectivity of the graph. To demonstrate this sensitivity, the figure below shows the result obtained using the worst γ value among those tested. In this specific instance, the algorithm fails to capture fine structural details; notably, the tripod is not correctly clustered and merges erroneously with the background. This visual evidence confirms that while Nyström is computationally efficient, proper hyperparameter tuning is essential to achieve accurate segmentation.



REFERENCES

[1] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[2] S. Ling, "Spectral Clustering, Graph Laplacian," Lecture notes, 2020.

[3] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the Nyström method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 214–225, 2004.

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, Portland, OR, 1996, pp. 226–231.

[5] E. Schubert, J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 19:1–19:21, Jul. 2017.