Practical Session 1

Spectral Clustering

Graphs in Machine Learning
MVA Master Program – ENS Paris-Saclay

michal.valko@inria.fr
daniele.calandriello@inria.fr
pierre.perrault@inria.fr
omar.darwiche-domingues@inria.fr

About this Session

In this practical session we will cover fundamental graph building techniques, and apply them to the Spectral Clustering problem. The session will be evaluated on a short written report and a final image segmentation implementation. During the TD we will implement all the necessary tools to do the segmentation and answer the report questions. All the code related to the TD must be submitted, to provide background for the image segmentation code evaluation.

1 Graph Construction

The file `generate_data.py` contains functions that will generate artificial data for the experiments, as described below. You can run the script to visualize the data.

Instructions

Dataset Types:

- N**-Blob**: Sample random points in $\mathbb{R}^2$ according to N Gaussian distributions with mean $\mu_i = [\cos(2\pi i/N), \sin(2\pi i/N)]$ and variance $\text{Diag}(\sigma^2)$.
- **Two Moons**: Sample random points shaped as two intertwined moons.
- **Point and Circle**: Sample random points from a concentrated Gaussian point in the middle and a wide circle around it.

A prerequisite to build a similarity graph is to define a similarity function to score the distance between nodes in the graph. For the rest of the session we will use an inverse exponential function as the similarity measure, controlled by the Euclidean distance:

$$d(x_i, x_j) = \exp\left\{-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}\right\} \tag{1}$$

The variance σ^2 controls the bandwidth of the similarity.

1.1 Implementation Tasks

In the file `build_similarity_graph.py`, complete the following:

- Write the code to build an ε-graph and a (OR) k-nn graph. Details are in the source code.
- Use `plot_similarity_graph` to visualize the graph for generated data. The function `plot_graph_matrix` in `utils.py` may be useful.
- Complete the function `how_to_choose_epsilon`. You may use `min_span_tree` in `utils.py`.

Questions

1. What is the purpose of the option parameter in `worst_case_blob`?
2. What happens when you change the generating parameter of `worst_case_blob` in `how_to_choose_epsilon`? What if the parameter is very large?
3. Using `plot_similarity_graph`, compare k-nn to ε graphs. When is it easier to build a connected graph using k-nn? When using ε graphs?

2 Spectral Clustering

In the file `spectral_clustering.py`, complete:

- `build_laplacian`
- `spectral_clustering`

Questions

1. Build a graph from `two_blobs_clustering` data, keeping the graph connected. Motivate your eigenvector choices and how you computed clustering assignments. Compare with built-in k-means.
2. Build a graph from `two_blobs_clustering`, but make the two components separate. How do you choose eigenvectors? Motivate your answer.

2.1 Adaptive Eigenvector Selection

Complete `spectral_clustering_adaptive`, which uses `choose_eig_function`.

Questions

1. Look at `find_the_bend`. Generate 4-blob data with $\sigma^2 = 0.03$. Plot the first 15 eigenvalues. Complete `choose_eig_function` to automatically choose the number of eigenvectors. The rule must adapt to actual eigenvalues.
2. Increase variance to $\sigma^2 = 0.20$ while plotting eigenvalues. Use `choose_eig_function`. Do you see any difference?
3. When building cluster assignments, did you use thresholding, k-means or both? When to use each?
4. What is another use of eigenvalue distributions during clustering, beside choosing eigenvectors?

2.2 Complex Structures

Complete:

- `two_moons_clustering`
- `point_and_circle_clustering`

Questions

1. Plot results using spectral clustering and k-means in `two_moons_clustering`. Notice differences? Explain considering graph structure.
2. In `point_and_circle_clustering`, compare spectral clustering using normal Laplacian L and random-walk regularized Laplacian L_{rw}. Notice differences? Explain considering graph structure.

2.3 Parameter Sensitivity

How did you choose parameters when building the graph? We'll compare clustering solutions while changing parameters. We use the Adjusted Rand Index [**?**] to evaluate assignments, which outputs values between 0 (unrelated) and 1 (equal).

Questions

1. Complete `parameter_sensitivity`, and generate a plot of the ARI index while varying one parameter in graph construction (ε or k). Comment on spectral clustering stability.
2. Without access to *true* labels, how could we evaluate clustering results?

3 Image Segmentation

Your final task is implementing image segmentation using spectral clustering. Complete `image_segmentation.py`.

Instructions

Pointers:

- Images are 50×50 pixels RGB in the data folder. The image is loaded as $50 \times 50 \times 3$ matrix, converted to 2500×3 where each pixel is an $\mathbb{R}^3$ vector in RGB space.
- Images can be segmented using colors. Build a graph based on color distance between pixels and cluster them.

The function `image_segmentation` will plot your segmentation. Colors might be arbitrary since clustering labels are arbitrary—separation correctness is what matters.

Questions

Final Questions:

1. Document your implementation choices. Well-written, well-commented code is sufficient. Include all related code in submission.
2. A full graph on 50×50 image has 50^2 nodes. Eigenvalue solving scales as $O(2^{34})$ (seconds to minutes on weak hardware). Full HD (1920×1080) scales as $O(2^{64})$ (about a month). A full graph on millions of nodes requires 1TB memory. Can you think of two simple techniques to reduce computational and memory cost?
3. Did you use `eig` or `eigs`? What's the difference? How do they scale to large graphs?