Foundations of Machine Learning
Assignment 1

Name: Aryan Prasad
Roll Number: DA25M007

September 21, 2025

Question 1: PCA and Kernel PCA

Dataset Overview

Theory: We are given a dataset with 1000 points in $\mathbb{R}^2$. The goal is to apply **Principal Component Analysis (PCA)** on this dataset, and then try out **Kernel PCA (KPCA)** with different kernels. After that, we compare the results and decide which kernel is most suitable.

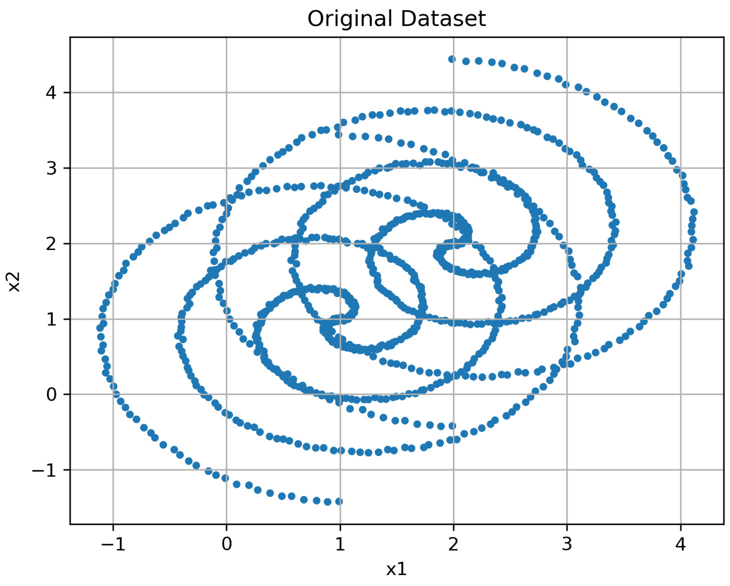

Figure 1: Original dataset scatter plot (1000 points in $\mathbb{R}^2$). The points form spiral-like curves.

(a) PCA Implementation

PCA is a simple but powerful technique for reducing dimensions. The main idea is to rotate the coordinate system so that the new axes (principal components) align with the directions of maximum variance in the data.

Steps I followed:

1. **Centering the data:** I subtracted the mean of each feature so that the dataset has mean zero.
$$X_c = X - \mu, \quad \mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

2. **Covariance matrix:** I calculated the covariance matrix to see how the two features vary together.
$$C = \frac{1}{n-1} X_c^T X_c$$

3. **Eigen decomposition:** I solved the eigenvalue problem:
$$Cv = \lambda v$$
where λ is the variance captured by eigenvector v.

4. **Sorting eigenvalues:** The eigenvalues were sorted in descending order to pick the directions with the most variance.

5. **Explained Variance Ratio (EVR):**
$$EVR_i = \frac{\lambda_i}{\sum_j \lambda_j}$$
This shows how much of the total variance is explained by each component.

6. **Projection:** Finally, the data was projected onto the new principal components.
$$Z = X_c V$$

Results:

- Eigenvalues: $\lambda_1 \approx 1.63, \ \lambda_2 \approx 0.86$
- Explained Variance Ratio (EVR): First PC $\approx 65\%$, Second PC $\approx 35\%$

This means the first component alone explains most of the variation in the dataset, while the second accounts for the remaining spread.

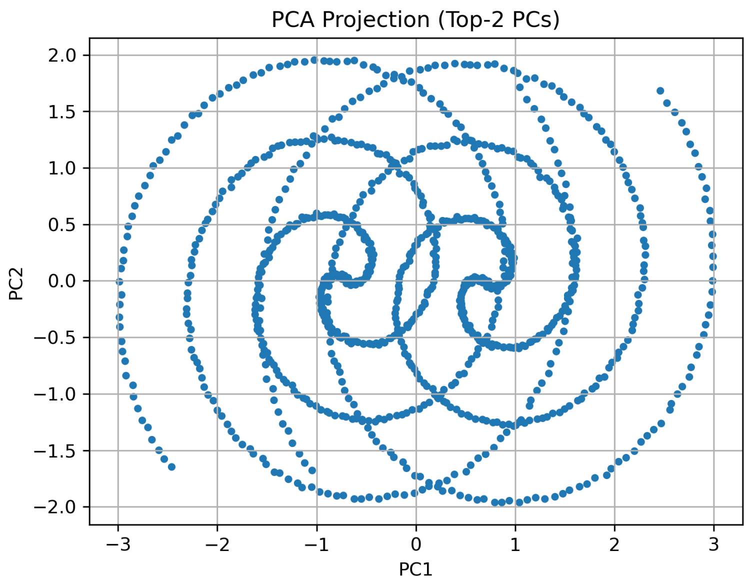

Figure 2: PCA projection onto top-2 principal components. The data is spread across the main variance directions.

(b) Kernel PCA Implementation

While PCA works well for linear patterns, it struggles when the data has nonlinear shapes like spirals or circles. To handle this, Kernel PCA uses a **kernel function** to compute similarity between points. This trick makes it look as if the data has been mapped into a higher-dimensional space where linear PCA can then be applied.

Kernels I used:

- **Linear Kernel:** $k(x, y) = x^T y$ Works the same as PCA since it is just the dot product.
- **Polynomial Kernel:** $k(x, y) = (\gamma x^T y + c_0)^d$ Can capture curved shapes depending on the degree d.
- **RBF Kernel:**
$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$
Here, σ controls how local the similarity is. A small σ focuses on very close neighbors, while a large σ smooths the data more globally.

Results and Observations:

Linear Kernel: The projection was essentially identical to PCA, which was expected.

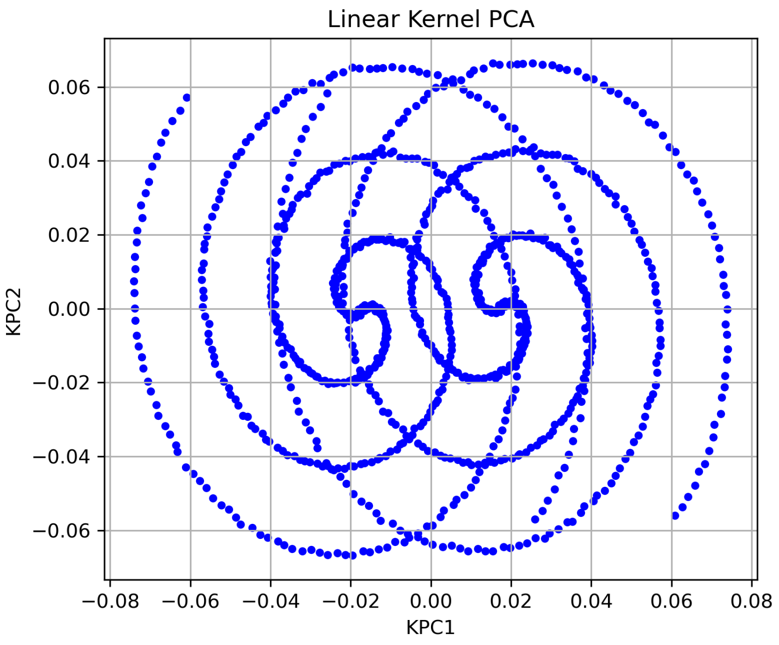

Figure 3: Kernel PCA with Linear Kernel. The result is the same as PCA.

Polynomial Kernel (degree 3): The data showed more curvature compared to linear PCA, but the plot looked a bit distorted and sensitive to the chosen degree.

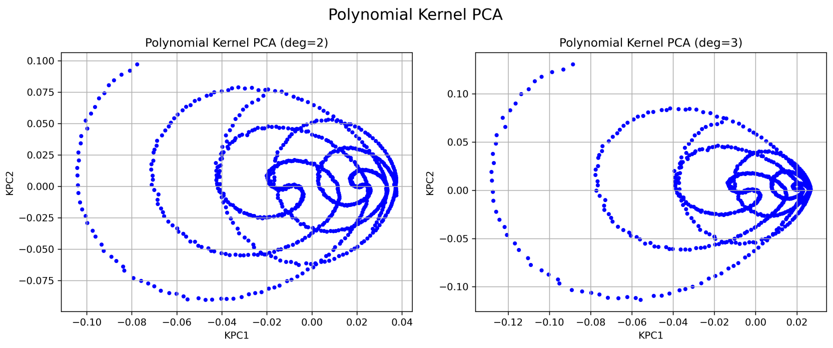

Figure 4: Kernel PCA with Polynomial Kernel (degree=3). Captures nonlinear shapes but is not very stable.

RBF Kernel ($\sigma = 5.0$): At this value of σ, the data became too smooth and much of the structure was lost. From experiments with smaller σ values (1.0–2.0), the data looked more unfolded and made better sense.

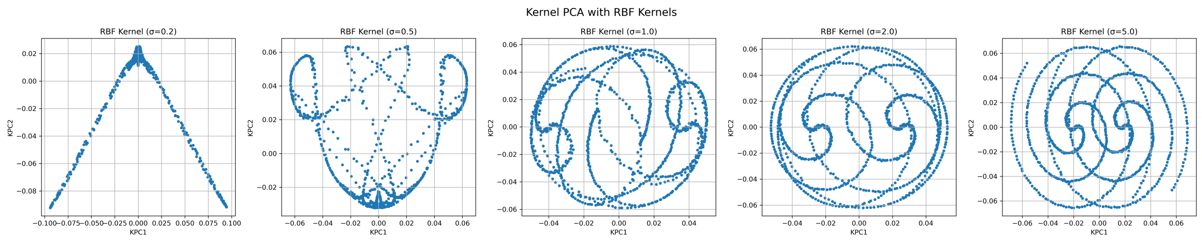

Figure 5: Kernel PCA with RBF Kernel ($\sigma = 5.0$). The structure is oversmoothed and not clearly separated.

(c) Best Kernel and Why

- The Linear kernel does not add anything new compared to PCA.
- The Polynomial kernel shows some nonlinear patterns but is sensitive to the degree value.
- The RBF kernel gave the best results. For medium σ values (1.0–2.0), it unfolded the spiral-like dataset and showed clear structure without breaking it apart or oversmoothing.

Answer: The **RBF kernel with σ between 1.0 and 2.0** worked best for this dataset.

Conclusion

- PCA reduced the dataset to new directions and showed that about 65% of the variance was explained by the first component.
- Kernel PCA was able to capture nonlinear patterns that PCA could not.
- Out of all kernels, the RBF kernel with a medium σ gave the clearest and most useful projection.

In summary, PCA is simple and effective for linear data, while Kernel PCA, especially with the RBF kernel, is much more powerful for data with curved or spiral structures.

Q2: Clustering on Synthetic Dataset

Dataset Overview

Theory: The dataset consists of points arranged in four concentric rings.. Ring-shaped clusters are non-linear and non-convex, so they require more advanced methods such as spectral clustering.

Clustering here aims to separate each ring into its own cluster. This provides a strong test case for comparing K-Means, Voronoi analysis, and spectral methods.

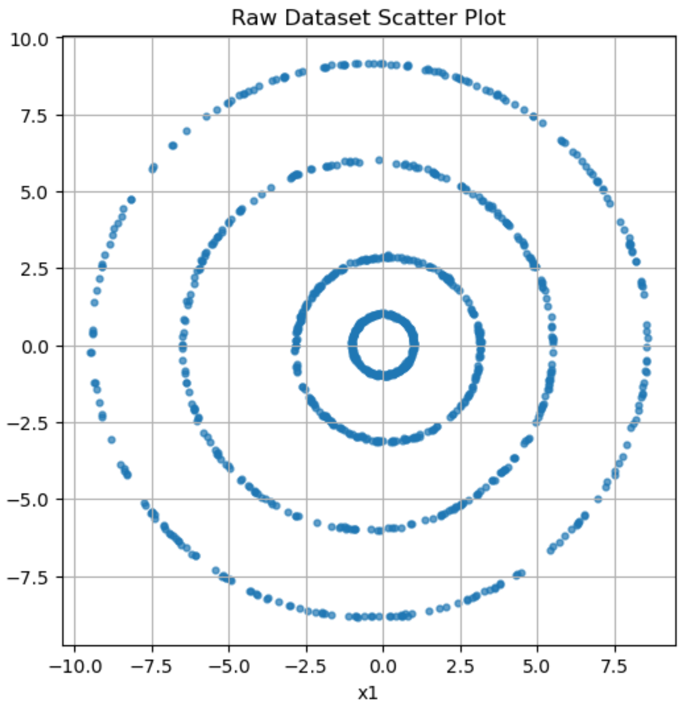

Figure 6: Initial synthetic dataset of concentric rings.

(a) K-Means with Multiple Initializations

Theory: K-Means is one of the most popular clustering methods. It aims to divide a dataset into k groups by minimizing the squared distance between points and the closest cluster center:

$$J = \sum_{i=1}^{n} \min_{1 \le j \le k} \|x_i - \mu_j\|^2.$$

The procedure alternates between two steps: 1. **Assignment:** Allocate each sample to its nearest center. 2. **Update:** Recalculate centers as the mean of assigned samples.

This loop continues until convergence (or when the decrease in J is negligible). A key limitation is that the outcome depends strongly on the starting positions of centers.

Results: We applied K-Means with $k = 4$ using five random seeds. Each experiment reports inertia vs iterations (left) and the final clusters (right).

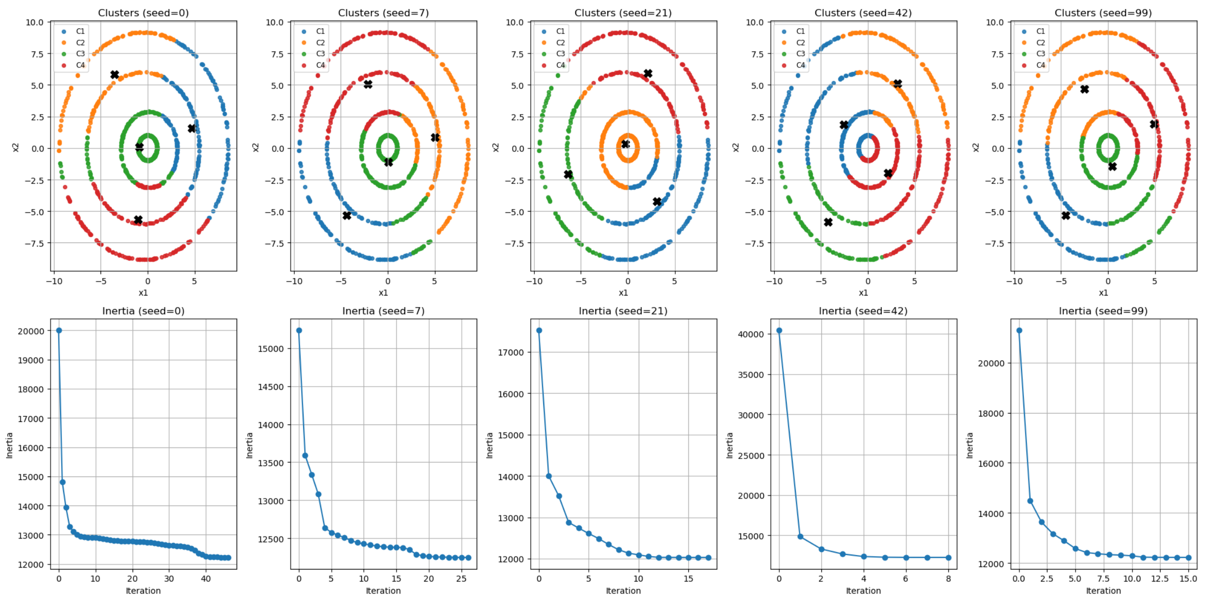

Figure 7: K-Means with different random seeds. Left: inertia curves, Right: cluster assignments.

Observations: - Inertia drops steeply at the start, confirming fast convergence. - Different seeds lead to different partitions, showing initialization sensitivity. - Clusters appear as radial wedges cutting across rings. - This occurs because K-Means enforces *convex Voronoi cells*, unsuitable for circular structures. - Works well for convex shapes, but fails here.

(b) Voronoi Regions for Different K

Theory: The regions produced by K-Means are Voronoi cells:

$$V_j = \{x \in \mathbb{R}^d : \|x - \mu_j\| \leq \|x - \mu_\ell\|, \forall \ell \neq j\}.$$

Since Voronoi cells are convex, they cannot represent curved boundaries like rings.

Results: We visualized Voronoi diagrams for $K = 2, 3, 4, 5$.

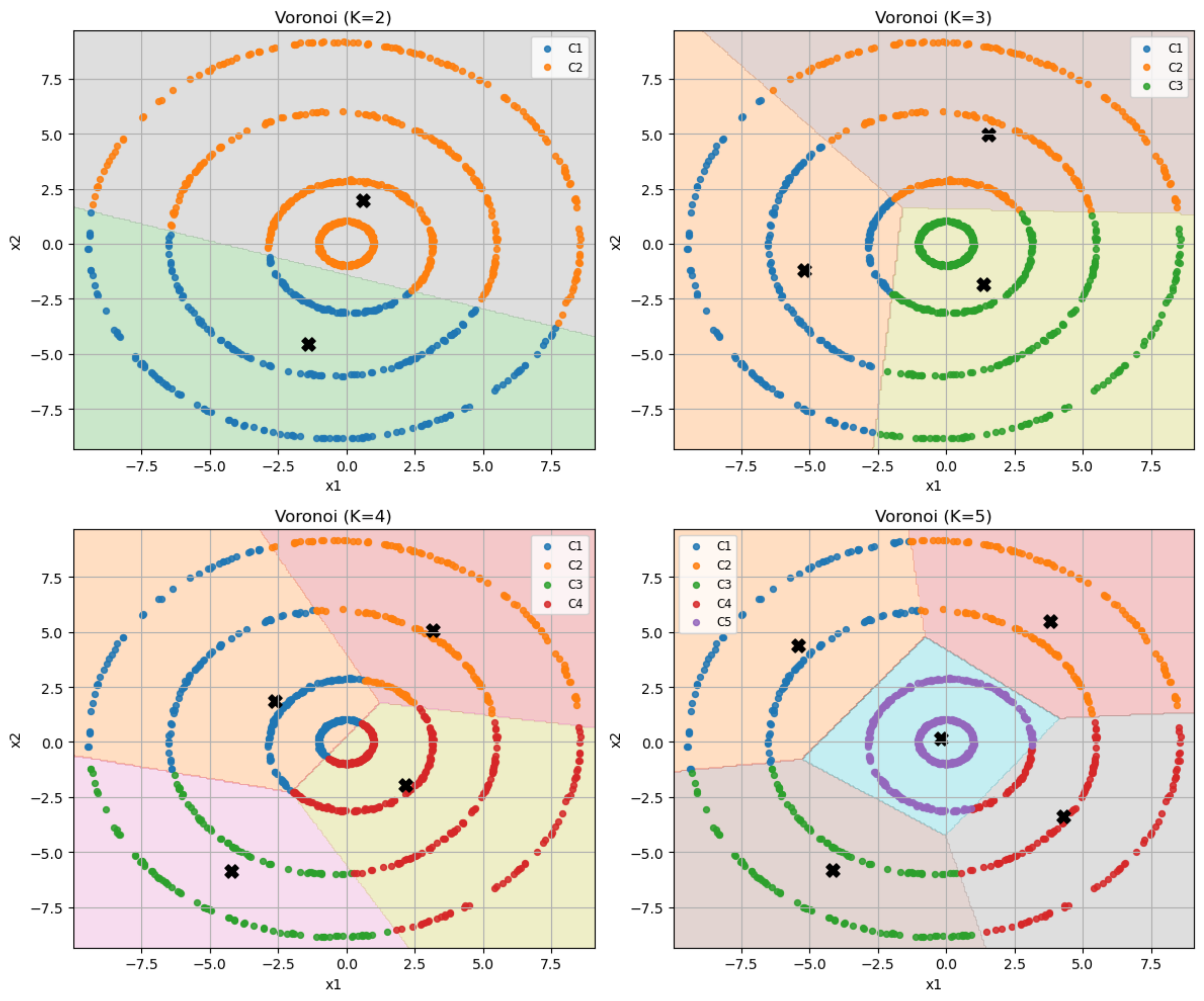

Figure 8: Voronoi regions for various K values.

Observations: - For $K = 2$ or $K = 3$, multiple rings collapse into the same cluster (underfitting). - At $K = 4$, the number of clusters matches the rings but slices them into wedges. - At $K = 5$, one ring is unnecessarily divided (overfitting). - Confirms Voronoi boundaries are too rigid.

(c) Spectral Clustering with K-Means

Theory: Spectral clustering solves this issue by embedding points in a new space defined by eigenvectors of a similarity matrix.

The similarity is measured with a Gaussian kernel:

$$W_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right).$$

From W, we build the degree matrix D and normalized Laplacian:

$$L_{\text{sym}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}.$$

Eigenvectors of L_{sym} (or the kernel matrix) uncover the hidden structure. After row normalization, running K-Means in this new space separates even non-convex sets.

Results: We tested both Kernel PCA and Laplacian methods for $\sigma = 0.1, 0.3, 0.5, 1.0$.

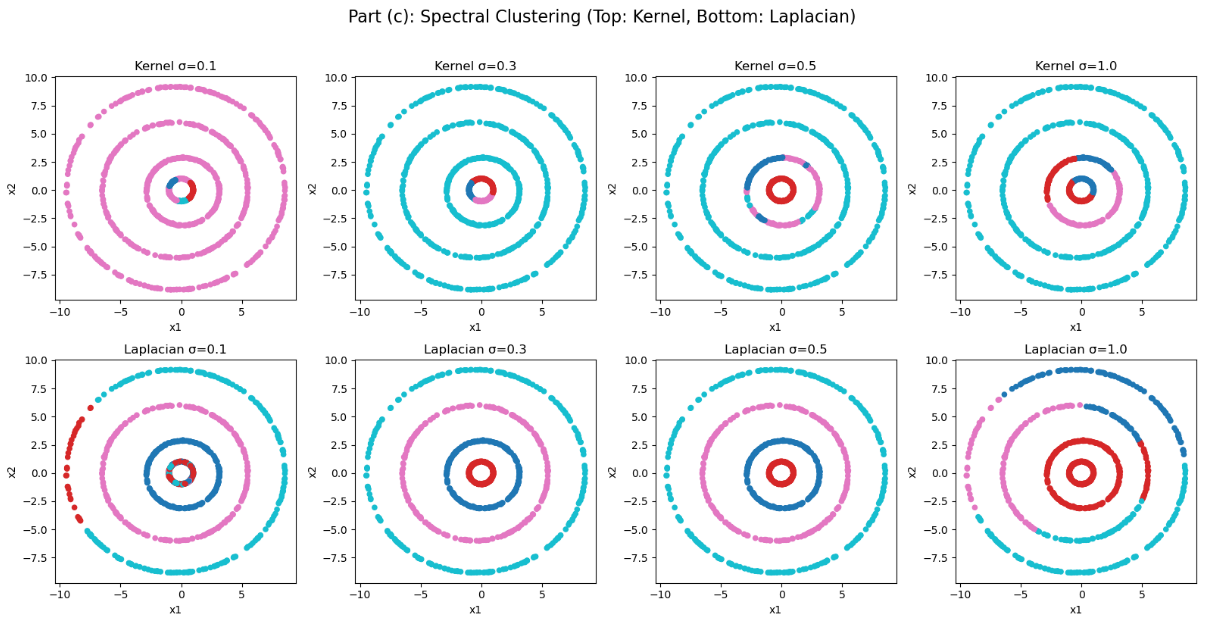

Figure 9: Spectral clustering results. Top: Kernel approach, Bottom: Laplacian approach.

Observations: - With $\sigma = 0.1$, Kernel PCA collapses, while Laplacian starts showing separation. - At $\sigma = 0.3$, Laplacian cleanly split the four rings but Kernal PCA doesn't. - With larger σ (0.5, 1.0), similarities blur and rings merge. - Laplacian proves more stable overall for this dataset.

Key Point: Spectral clustering embeds the data into a space where rings become linearly separable, unlike plain K-Means. '

(d) Alternative Mapping via Eigenvectors

Theory: Instead of running K-Means on the embedding, clusters can be assigned using the maximum component of the eigenvector representation:

$$\ell(i) = \arg \max_{1 \le j \le k} v_{ji}.$$

This avoids K-Means but is less reliable since eigenvectors may not clearly define cluster boundaries.

Results: Kernel and Laplacian versions were tested with $\sigma = 0.1, 0.3, 0.5, 1.0$.

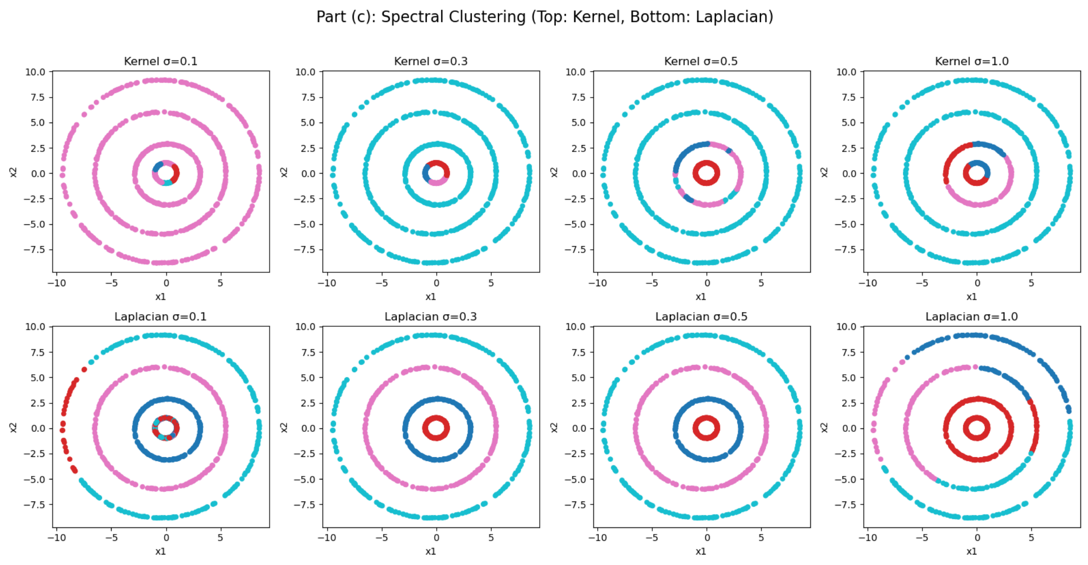

Figure 10: Alternative mapping. Top: Kernel method, Bottom: Laplacian method.

Observations: - Kernel mapping collapses to 1–2 groups for all σ. - Laplacian mapping works better: clean rings at $\sigma = 0.3$, noisy at 0.1, merging at higher values. - Still weaker than spectral clustering with K-Means.

Final Insights

- K-Means struggles with non-convex data and depends heavily on initialization.
- Voronoi analysis confirms its geometric rigidity.
- Spectral clustering, especially Laplacian with $\sigma \approx 0.3$, successfully recovers all rings.
- Direct eigenvector mapping is less effective, showing why the K-Means step is valuable.

Conclusion: The most effective method for this dataset is **Laplacian spectral clustering with $k = 4$ and $\sigma = 0.3$**, which consistently extracts all four rings and is more reliable than Kernel PCA or plain K-Means.