URL Coursework 1: Clustering
Agglomerative clustering via maximum incremental path integral [1]

Bruno Sánchez Gómez

March 19, 2025

Abstract

Your abstract here.

1 Introduction

Introduce the study of this project.

2 Path Integral Clustering

Path Integral Clustering (PIC)[1] is an agglomerative clustering approach that leverages the structural properties of a data graph to measure cluster stability. By integrating over all possible paths within a cluster, PIC quantifies how strongly connected the members are. A key insight of PIC is treating each cluster as a dynamical system, with its samples as states — a concept borrowed from statistical and quantum mechanics. The path integral then serves as a structural descriptor measuring the stability of this dynamical system. Clusters that exhibit a large path integral are deemed stable, and the incremental increase in the path integral after merging two clusters is used as an affinity measure. This incremental path integral can be computed through a closed-form exact solution with linear time complexity relative to the maximum cluster size. This section describes both the theoretical algorithm and its practical implementation.

2.1 PIC Algorithm

The PIC algorithm consists of the following key steps (this is a summarized version; the full description can be found with more detail in [1]):

Graph Construction. Given a dataset $\{x_i\}_{i=1}^n$, a directed graph $G = (V, E)$ is constructed where each data point corresponds to a vertex. For a pair of points, the weight of the edge from x_i to x_j is defined as

$$w_{ij} = \begin{cases} \exp\Big(-\frac{\mathrm{dist}(i,j)^2}{\sigma^2}\Big), & \text{if } x_j \in \mathcal{N}_i^K, \\ 0, & \text{otherwise,} \end{cases}$$

where $\mathrm{dist}(i,j)$ is the distance between x_i and x_j, $\mathcal{N}_i^K$ is the set of K nearest neighbors of x_i, and σ^2 is calculated as follows:

$$\sigma^2 = \frac{\sum_{i=1}^n \sum_{x_j \in \mathcal{N}_i^3} \mathrm{dist}(i,j)^2}{3n(-\ln a)}$$

The transition probability matrix P is then computed by normalizing the rows of the weight matrix:

$$p_{ij} = \frac{w_{ij}}{\sum_j w_{ij}}.$$

Initial Clustering. A simple nearest-neighbor merging is applied to form initial clusters. Each data point is merged with its nearest neighbor, and the connected components of the resulting graph are used as the starting clusters.

Path Integral Computation. For any cluster C, the stability is quantified by its path integral:

$$S_C = \frac{1}{|C|^2}\,\mathbf{1}^T\Big(I - zP_C\Big)^{-1}\mathbf{1},$$

where P_C is the submatrix of P corresponding to the vertices in C, $z \in (0,1)$ is a weighting parameter that favors short paths, and $\mathbf{1}$ is a vector of ones. The path integral represents the total contribution of all paths (of all lengths) within the cluster and can be interpreted as the probability that a random walk starting in C remains in C.

Affinity Measurement and Cluster Merging. The affinity between two clusters, say C_a and C_b, is measured by the incremental path integral:

$$\mathcal{A}_{C_a,C_b} = \Big(S_{C_a|C_a\cup C_b} - S_{C_a}\Big) + \Big(S_{C_b|C_a\cup C_b} - S_{C_b}\Big),$$

where $S_{C|C_a\cup C_b}$ is the conditional path integral computed on the union of clusters, with the contributions only from vertices originally in C. A high affinity indicates that merging the clusters significantly increases the number of intra-cluster paths; therefore, creating a more stable cluster. The algorithm iteratively merges the pair of clusters with the highest affinity until the desired number of clusters is achieved.

2.2 PIC Implementation

The implementation of PIC is designed for computational efficiency and ease of integration. Key aspects of the implementation are described below:

Graph Construction via Nearest Neighbors. The implementation first computes the 3-nearest neighbors of each point to robustly estimate the scale parameter σ^2. A subsequent K-nearest neighbor search then constructs the weighted graph and the transition probability matrix P.

Efficient Path Integral Evaluation. Rather than computing the inverse of $(I - zP_C)$ explicitly—which can be computationally expensive—the algorithm solves the linear system

$$(I - zP_C)y = \mathbf{1}$$

using numerical solvers (e.g., via `numpy.linalg.solve`). This approach takes advantage of the sparsity of P_C and reduces the complexity to be linear in the size of the cluster.

Incremental Cluster Merging Using a Heap. A priority queue (implemented as a max-heap) is used to efficiently identify the pair of clusters with the highest affinity. After merging two clusters, only the affinities involving the new cluster are updated and inserted into the heap. This strategy minimizes unnecessary computations and allows the algorithm to scale to larger datasets.

Modular Code Structure. The PIC algorithm is implemented as a Python class conforming to the `sklearn` API. Key functions are modularized as follows:

- `_build_graph`: Constructs the weighted graph and computes the transition probability matrix.
- `_initial_clusters`: Forms initial clusters using nearest neighbor merging.
- `_compute_S` and `_compute_S_cond`: Compute the path integral for a cluster and its conditional counterpart for merged clusters.
- `_merge_clusters`: Iteratively merges clusters based on the incremental path integral, using a heap to manage candidate pairs.
- `fit`: Fits the PIC model to the data and computes the clusters.
- `predict`: Assigns new samples to the nearest cluster center.
- `fit_predict`: Fits the PIC model to the data and returns the cluster labels.

These design decisions ensure that the algorithm remains both theoretically robust and practically efficient.

Overall, the PIC implementation balances the theoretical formulation of the path integral with practical considerations for computational efficiency, making it suitable for clustering tasks on datasets with complex, manifold-like structures.

3 Experiments

This section presents the experimental evaluation of the Path Integral Clustering (PIC) algorithm compared to various state-of-the-art clustering methods. We assess performance on both synthetic and real-world imagery datasets, following the experimental framework described in the original paper[1]. Then, we present some criticism and issues encountered during the experiment replication process. Finally, we perform some additional experiments to further evaluate PIC's performance.

3.1 Synthetic Datasets

We recreated the three synthetic datasets introduced in [1] to visually demonstrate PIC's effectiveness on data with complex structures.

Figure 1 shows the clustering results on these synthetic datasets.

The results demonstrate PIC's ability to handle complex data structures. Particularly noteworthy is PIC's performance on Dataset 1, where it successfully identified both the dense clusters and the circular pattern despite the presence of noise. For Dataset 2, PIC effectively captured the sinusoidal and non-convex patterns, outperforming traditional algorithms that typically prefer convex clusters. Dataset 3 showcases PIC's robustness to noise, accurately separating the two main clusters from the surrounding noise.

3.2 Imagery Datasets

We evaluated PIC and 11 other clustering algorithms on three widely used image datasets:

1. **MNIST**: Handwritten digits (0-4), with 5,139 samples and 784 dimensions (28×28 pixels).
2. **USPS**: Handwritten digits (0-9), with 9,298 samples and 256 dimensions (16×16 pixels).
3. **Caltech-256**: Reduced to six classes (hibiscus, ketch-101, leopards-101, motorbikes-101, airplanes-101, and faces-easy-101), with 600 samples and 4,200 dimensions (60×70 grayscale images).

NMI	**MNIST**	**USPS**	**Caltech-256**
PIC	**0.940**	0.835	**0.653**
k-med	0.318	0.553	0.315
A-link	0.408	0.139	0.313
S-link	0.002	0.002	0.019
C-link	0.539	0.374	0.395
AP	0.426	0.525	0.492
NCuts	0.807	0.772	0.589
NJW	0.898	0.784	0.529
CT	0.634	0.439	0.181
Zell	0.913	**0.846**	0.343
C-kernel	0.780	0.768	0.521
D-kernel	0.903	**0.846**	0.508

Table 1: Normalized Mutual Information (NMI) scores for all algorithms on image datasets. Higher values indicate better performance. Bold indicates best performance.

Looking at the results in Tables 1 and 2, we observe that:

- **MNIST**: PIC achieved the highest NMI (0.940) and lowest CE (0.016), significantly outperforming other methods. This suggests that PIC effectively captures the intrinsic manifold structure of the handwritten digits.
- **USPS**: PIC performed well with an NMI of 0.835, though slightly behind the Diffusion kernel (D-kernel) and Zell methods which both achieved an NMI of 0.846. In terms of CE, D-kernel had the best performance (0.132), followed by Zell (0.197) and PIC (0.269). These results still show strong performance for PIC.

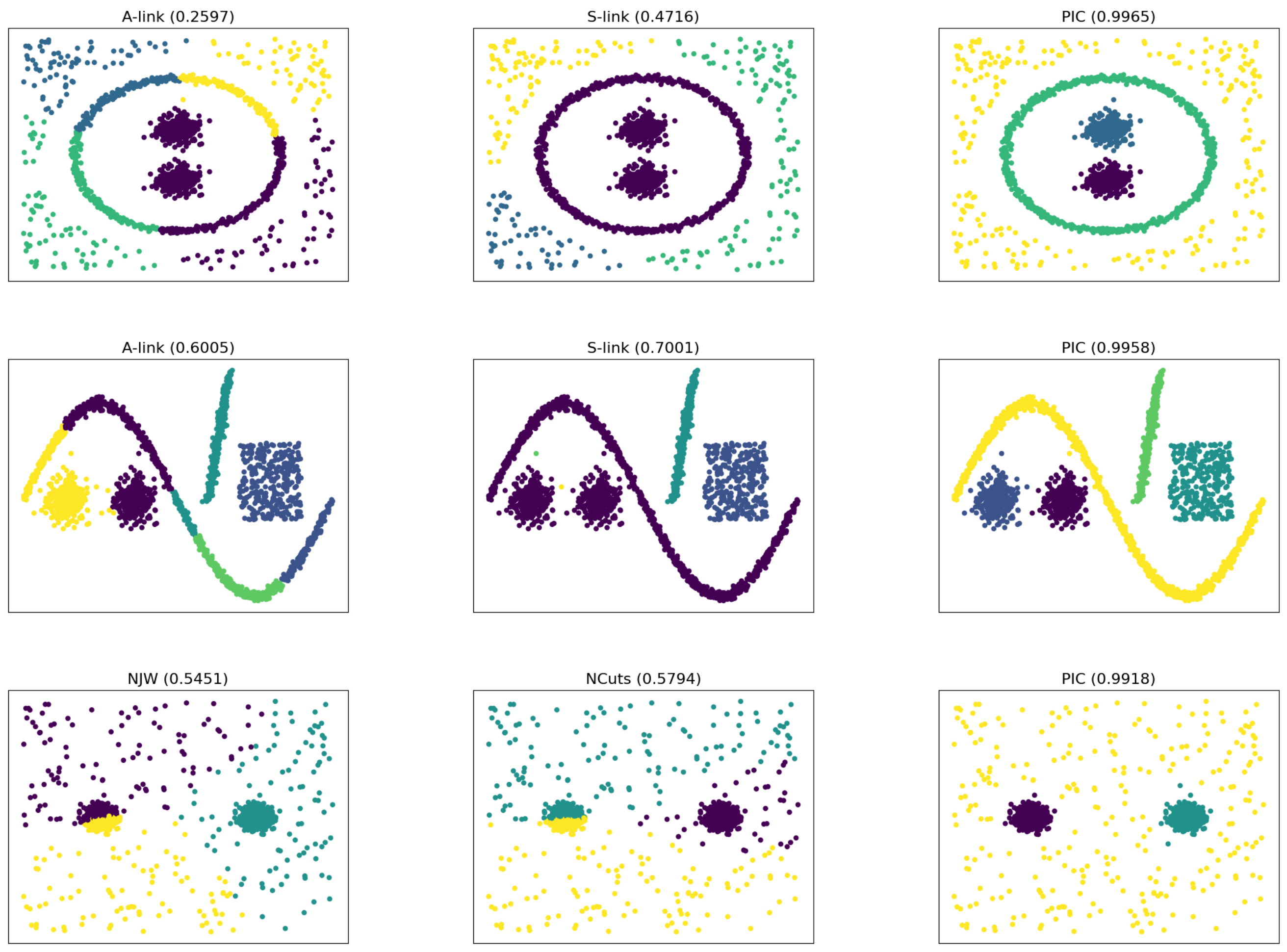

Figure 1: Clustering results on the three Synthetic Datasets (NMI scores in parentheses).

CE	**MNIST**	**USPS**	**Caltech-256**
PIC	**0.016**	0.269	0.307
k-med	0.534	0.373	0.607
A-link	0.573	0.778	0.665
S-link	0.779	0.833	0.828
C-link	0.280	0.601	0.507
AP	0.960	0.934	0.705
NCuts	0.115	0.356	0.328
NJW	0.033	0.269	**0.290**
CT	0.493	0.615	0.747
Zell	0.027	0.197	0.680
C-kernel	0.129	0.269	0.368
D-kernel	0.029	**0.132**	0.315

Table 2: Clustering Error (CE) scores for all algorithms on image datasets. Lower values indicate better performance. Bold indicates best performance.

- **Caltech-256**: PIC significantly outperformed all other methods with an NMI of 0.653. For CE, NJW had the lowest value (0.290), followed closely by PIC (0.307). This demonstrates PIC's ability to handle higher-dimensional image data with complex visual patterns.

Our results largely align with those reported in the original paper, with PIC consistently performing as one of the top methods across datasets. However, we observed some differences:

1. On the USPS dataset, our implementation shows D-kernel and Zell slightly outperforming PIC, whereas the original paper reported PIC as the best method. This discrepancy might be due to differences in the dataset composition (9,298 samples in our case versus 11,000 mentioned in the paper), or in each of our custom implementations (since these algorithms are not supported by stardard libraries).
2. For Caltech-256, we achieved similar relative performance between methods, though our absolute scores differ from the paper, likely due to different preprocessing approaches.

3.3 Issues Encountered

Several challenges were encountered during the attempt to replicate the original paper's experiments with the highest fidelity possible:

1. **Synthetic Dataset Generation**: The original paper did not provide clear guidelines for synthetic dataset generation. Considerable tuning was required to create datasets in which the PIC algorithm exhibited the behaviors described in the paper.
2. **Algorithm Implementation**: Implementing all 11 comparison algorithms was challenging, requiring adaptation of existing libraries and development of custom implementations, since not all of them are supported by stardard Python libraries.

3. **Dataset Availability**: Two datasets mentioned in the original paper were not available: FRGC-T requires restricted access, and PubFig is no longer publicly available.

4. **Dataset Discrepancies**: The USPS dataset contained 9,298 samples instead of the 11,000 mentioned in the paper.

5. **Preprocessing Ambiguity**: For Caltech-256, the paper stated a dimensionality of 4,200 but did not specify how images of different sizes were processed. We adopted a 60×70 grayscale representation.

Despite these challenges, our implementation successfully reproduced the main findings of the original paper, confirming PIC's effectiveness for clustering tasks, especially on datasets with complex manifold structures.

3.4 Additional Experiments

To further evaluate PIC's performance, we conducted some additional experiments. First, we gathered time-related information from the imagery datasets experiments in order to perform a scalability analysis.

3.4.1 Scalability Analysis

The original paper states that the PIC algorithm scales linearly with the number of clusters. However, it does not mention the algorithm's scalability with respect to the number of samples or dimensions. To investifate this, we recorded the runtime of PIC on each of the three image datasets, which have different number of samples, clusters and dimensions. The results are shown in Table 3.

	MNIST	**USPS**	**Caltech-256**
# of samples	5,139	9,298	600
# of clusters	5	10	6
Dimensionality	784	256	4,200
PIC Runtime (s)	256.1	854.7	1.5

Table 3: Runtime of PIC on different datasets with varying sizes and dimensionalities.

The results show that PIC's runtime does not seem to be affected by the dimensionality of the dataset in any significant way, since the Caltech-256 dataset has the highest number of dimensions and yet has the lowest runtime by a wide margin. However, the runtime does greatly increase with the number of samples, as seen in the MNIST and USPS datasets. The number of clusters could also have an impact, but it does not seem to be as significant as that of the number of samples.

4 Conclusion

Your conclusion here.

References

[1] Wei Zhang, Deli Zhao, and Xiaogang Wang. Agglomerative clustering via maximum incremental path integral. *Pattern Recognition*, 46(11):3056–3065, 2013.