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

Bruno Sánchez Gómez

March 18, 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. 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 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

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

where $\text{dist}(i, j)$ 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:

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

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

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

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

Experiments and difficulties.

3.1 Synthetic Datasets

3.2 Imagery Datasets

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

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

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.