Agglomerative clustering via maximum incremental path integral: an implementation

Unsupervised and Reinforcement Learning

April 13, 2025

Bernat Comas i Machuca

Contents

1 Introduction to the problem

The paper Agglomerative clustering via maximum incremental path integral (2013) [1] presents a new agglomerative clustering algorithm that computes cluster similarity based on path graphs that capture the data structure. To make this computation each cluster is treated as a dynamic system and the algorithm measures stability using the Path Integral, a concept extracted from statistical mechanics and quantum mechanics. The algorithm decides how to merge clusters based on how much their stability changes, which results in an efficient solution with linear time complexity.

To define the structural descriptors that allow to compute this concept of similarity, a neighborhood graph is created. This allows it to avoid relying as much in pairwise distances (only used for graph initialization), and instead define the affinity between clusters by how much their stability changes when merged, which is measured through the path integral.

This has several advantages:

- Works well for data lying on low-dimensional manifolds.
- It does not rely on approximation or eigen-decomposition, making it more robust to noise.
- It does not make assumptions on data distribution, offering better generalization.
- Performs well on multi-scale data
- It is efficiently computed by the presented algorithm in linear time complexity

Over the course of this project, we are going to implement the Path Integral Clustering algorithm in an efficient way, and then we are going to compare it to other well-known clustering algorithms. In this section we are going to describe the algorithm, section 2 contains a description of the implementation , in section 3 the algorithm experiments and comparison is performed, and finally the last section is devoted to describing the conclusions of the project.

1.1 Cluster initialization

Because we are working with an Agglomerative Clustering algorithm, we have to define how will the initial clusters be formed. We could initialize the samples each in its own cluster and start the iteration, but instead the paper proposes to use nearest neighbor merging.

The algorithm of nearest neighbor merging consists of having each sample in its own cluster together with its nearest neighbor and then, the clusters that share samples are merged. This results in a number of initial clusters that is at least half the number of samples.

1.2 Building the structural graph

We define the structural graph G as the directed graph where each initial sample X is a node and E the set of edges connecting them. Two samples are connected by an edge if they are in its K closest neighbors (for a predefined graph).

Then, we compute the weighted adjacency matrix W, which contains the pairwise similarities between each pair of samples. Note that we are only computing similarities between each point and its K closest neighbors, the rest of similarities will be 0. Therefore, each element w_ij) of the matrix W is defined as shown in Equation 1.

$$w_{ij} = \begin{cases} \exp\left(\frac{-\text{dist}(i,j)^2}{\sigma^2}\right), & \text{if } \mathbf{x}_j \in \mathcal{N}_i^K, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Where K is a free parameter to be set and σ^2 is estimated by Equation 2.

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

Where a is a parameter to be set.

We define the transition probability matrix P as the one-step transition probability from vertex i to vertex j, and we compute it using Equation 3.

$$\begin{aligned} P &= D^{-1}W; \\ d_{ii} = \sum_{j=1}^{n} w_{ij} \text{ such that } &\sum_{j=1}^{n} p_{ij} = 1 \end{aligned} \tag{3}$$

1.3 Affinity measure: Path integral

The path integral of a cluster is computed by summing the paths in the cluster on the directed graph G, weighted by transition probabilities in P.

The path integral used is a discretization of the one seen in quantum mechanics, and is defined as a sum of the weighted contributions of all the paths in the cluster C, divided by the number of samples belonging to the cluster. This is called the path integral descriptor of a cluster C_a, which we call S_{C_a}, and is defined in Equation 4. 4.

$$S_c = \frac{1}{|C|^2} \sum_{\gamma \in \Gamma_c} \Theta(\gamma) \tag{4}$$

Where, Γ_c is the set of all paths in C and $\Theta(\gamma)$ the weight of a path.

We also define the conditional path integral descriptor $S_{C_a|C_a \cup C_b}$ between two clusters S_{C_a} and S_{C_b} as the path descriptor of the clustering resulting from the merge but only taking into account the paths that have starting and ending vertices in C_a. We compute the conditional path integral descriptor using Equation 5.

$$S_{C_a|C_a\cup C_b} = \frac{1}{|C_a|^2}\mathbf{1}_{C_a}^T(\boldsymbol{I} - z\boldsymbol{P}_{C_a\cup C_b})^{-1}\mathbf{1}_{C_a} \tag{5}$$

Now, we can compute the affinity A_{C_a,C_b} of the merging of two clusters C_a and C_b by the increment in the path integral descriptor shown in the previous equations 4 and 5, as can be seen in Equation 6. In addition to the path integral of each cluster separately S_{C_a} and S_{C_b}, the computation of the affinity uses the conditional path integral descriptor $S_{C_a|C_a\cup C_b}$, which computes the resulting path descriptor of the merge but only the ones that have starting and ending vertices in C_a.

$$A_{C_a,C_b} = (S_{C_a|C_a\cup C_b} - S_{C_a}) + (S_{C_b|C_a\cup C_b} - S_{C_b}) \tag{6}$$

1.4 Agglomerative algorithm

The input to our problem is a set of sample vectors together with the target number of clusters (that has to be predefined).

The algorithm starts by building the graph, weighted adjacency matrix W, and using it to obtain the transition probability matrix P. Then, cluster initialization is run to obtain the initial clusters.

The iterative part consists of merging the two most affine clusters we have until the number of clusters is the target one. To do so, we use the precomputed affinities between all pairs of clusters. The pseudocode of the algorithm can be seen in Algorithm 1.

Algorithm 1 Path Integral Clustering Algorithm

1: **INPUT** The set of n points to cluster $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, and the target number of clusters n_T.
2: Build the graph G with k-nearest-neighbors and compute its weighted adjacency matrix $\mathbf{W}$.
3: Get the transition probability matrix $\mathbf{P}$
4: Run nearest neighbour merging to form n_c initial clusters $\mathcal{C}_c = \{\mathcal{C}_1, \ldots, \mathcal{C}_{n_c}\}$.
5: **while** $n_c > n_T$ **do**
6: Search two clusters $\mathcal{C}_a$ and $\mathcal{C}_b$, such that

$$\{\mathcal{C}_a, \mathcal{C}_b\} = \arg\max_{\mathcal{C}_a,\mathcal{C}_b\in\mathcal{C}_c} \mathcal{A}_{\mathcal{C}_a,\mathcal{C}_b}$$

where $\mathcal{A}_{\mathcal{C}_a,\mathcal{C}_b}$ is the affinity measure between $\mathcal{C}_a$ and $\mathcal{C}_b$.
7: $\mathcal{C}_c \leftarrow \{\mathcal{C}_c \setminus \{\mathcal{C}_a, \mathcal{C}_b\}\} \cup \{\mathcal{C}_a \cup \mathcal{C}_b\}$
8: $n_c \leftarrow n_c - 1$
9: **end while**
10: **OUTPUT** $\mathcal{C}_c$

2 Description of the implementation

In this section we are going to describe the particularities of the implementation of the Path Integral Clustering Algorithm. The first section goes over the implementation

techniques recommended by the paper, and the second one contains the extra decisions made during the implementation.

During the implementation, efficiency was our priority. The full implementation of the algorithm is contained in the files *pic.py*, *nearest_neighbour_init.py* and *path_integral.py*. The rest of files are helpers that have contributed to the results obtained but are not part of the implementation itself.

2.1 Efficient computation of the path integral

The main ideas laid out by the paper about the implementation are on the computation of the path integral between two clusters. They use Theorem 7 to avoid computing a matrix inverse but instead solve a linear system.

(7)

2.2 Implementation decisions

Because we want our algorithm to run in linear time, we need to precompute everything we can, and that includes not only the graph and its transition matrices, but also the affinity between each pair of clusters. Because as can be seen in 1 the decision on the clusters to merge is an argmax, we need to have computed the affinities between all pairs of clusters to compare them.

In addition to this, we are using a heap to store all the cluster pairs and their affinities for fast extraction in runtime. This brought us to use a set to store the clusters that are active in a moment, to avoid having to traverse the heap to remove all pairs of clusters that contains one of the merged ones at each iteration. This means that at every iteration we are flagging the merged clusters as inactive, computing the affinities between the new cluster and all the previous ones, and adding them to the heap.

Due to the nature of our algorithm, we only implement the fit-predict method. This is because the algorithm does not learn from data but instead applies the same process to the samples to cluster them independently from the data it has seen.

3 Experiments

In this project, we will conduct experiments comparing different clustering algorithms on several datasets. The algorithms under evaluation include Affinity Propagation, Complete Link, Average Link, Simple Link, Zell, and of course, our implementation of PIC. Each algorithm will be tested on four datasets: MNIST, USPS, Breast Cancer (Wisconsin), and a synthetic random dataset with 1000 samples and 50 features. The goal is to analyze the performance and effectiveness of these algorithms in terms of clustering quality and efficiency across different types of data.

The reason of selecting these algorithms to be compared is because we have only found implementations for the Affinity Propagation, Complete, Average and Simple Links, and to be able to compare it to some of the algorithms that get the most similar results in the paper we have manually implemented the diffusion Kernel (D-kernel) and Zeta function clustering (Zell), that get some of the best results in the paper after the PIC.

The metrics we will be using to compare the methods are the ones that are used in the paper: Normalized Mutual Information score and Clustering error, and as both of these are external metrics we have extended it with some internal ones, by the addition of Silhouette, Davies-Bouldin and Calinski-Harabasz. We have selected these three internal metrics because they are the ones recommended as internal criteria for a wide range of situations by the "Clustering evaluation and validation" chapter of the material of the Unsupervised Learning course [2].

The datasets chosen are some of the ones that are used in the paper: MNIST and USPS. We opted not to use the FRGC ver2.0, PubFig, and Caltech datasets for this study due to their large size, which would require significant computational resources and time for processing. Instead, we have decided to incorporate the Breast-cancer wisconsin dataset because it is a well-known benchmark in the medical domain, containing a high-dimensional, real-world classification and being widely used. We don't have access to the synthetic datasets that are used in the paper, and for this reason we have implemented our own. With our dataset, we will also study how does structural and gaussian noise affect our implementation of the clustering algorithm. Our synthetic dataset has 1000 samples, 10 features and 10 target clusters. We are going to do the same study as they did by introducing gaussian noise and structural noise.

- TODO: Table containing dataset size and information - BC Wisconsin: Biclass, as many merges as possible - Synthetic: All param combinations are repeated 20 times

Table 1: Statistics of used datasets.

Dataset	**USPS**	**MNIST**	**BC-Wisconsin**	**Synthetic**
No. of samples	11 000	5 139	569	1000
No. of clusters	10	5	2	10
Min. cluster size	1100	980	212	-
Max. cluster size	1100	1135	357	-
Noise added	No	No	No	Yes
Dimensionality	256	784	30	10

3.1 Noise analysis

3.2 Algorithm comparison

...

4 Conclusions

Comment about the claims of the paper, reproducibility & issues encountered.

5 Bibliography

[1] Wei Zhang et al., *Agglomerative clustering via maximum incremental path integral*, 2013 (Link)

[2] Javier Béjar, URL - 2025 Spring Term *Clustering evaluation and validation*