Improving Quake Fault Reconstruction Algorithms

Data Science Lab, Spring 2024

Frederieke Lohmann, Sam Houliston

Supervised by
Dr. Men-Andrin Meier

Coaches:
Daniil Dmitriev, Prof. Fanny Yang

June 5, 2024

Abstract

Computational seismology is thriving, leveraging advances in data science and recent enhancements in instrumentation to measure seismic events. Despite the availability of larger and higher-resolution datasets, the overarching goal of understanding earthquake triggering mechanisms remains hindered by the scarcity of labeled real data and adequate synthetic datasets for quantitative algorithm evaluation. Moreover, standard clustering algorithms used to identify fault lines from a seismic catalog underperform on seismic data, and specialized methods are still in their early stages. To address these challenges, we introduce a realistic synthetic seismic catalog generator and propose Probabilistic Agglomerative Clustering Favoring Planarity (PACP), an extension to the probabilistic fault clustering algorithm published by Kamer et al.. We apply PACP to generated synthetic datasets and to a real dataset from the Bedretto Underground Laboratory for Geosciences and Geoenergies. We observe a higher Adjusted Rand Index of the clusterings obtained with PACP on the synthetic data and a qualitatively better clustering on the real data compared to the baseline method of Kamer et al..

1 Introduction

Understanding the triggering mechanism of earthquakes has garnered a great interest in computational seismology for both academic research and practical applications (Plesch et al., 2007). Advances in instrumentation with a wider attention to document seismic events has enabled the analysis of larger and more complete seismic catalogs (Ouillon and Sornette, 2011). Such catalogs comprise 3-dimensional (3D) geographic coordinates of earthquake events. Patterns in these 3D point clouds can be revealed to distinguish fault lines - fractures between two blocks of rock, whose sudden relative movement is a mechanism of pressure relief that can induce earthquakes (Ohnaka, 2013). Properly identifying fault lines in seismic catalogs while interesting in and of itself, is key to understand properties of surrounding material, the architecture of the overall fault network, and eventually understand their risk of further propagation and subsequent quakes (Plesch et al., 2007).

The increasing availability of seismic data, coupled with growing interest and advancements in machine learning and algorithmic analysis, has expanded the repertoire of methods for fault detection and characterization (Kong et al., 2018). Despite these advancements, challenges persist in fault detection and characterization. One significant challenge is the scarcity of labeled data and the unavailability or inadequacy of synthetic datasets to quantitatively compare algorithms. Standard clustering algorithms often prove ineffective in delineating fault lines (Ouillon and Sornette, 2011), while specialized methods tailored for seismic data are still in their nascent stages and may falter even on basic cases we identify.

In response to these challenges, we introduce a realistic synthetic seismic catalog generator, we re-implement a promising algorithm recently published by Kamer et al. (2020) in python and propose an improved method, Probabilistic Agglomerative Clustering Favoring Planarity (PACP), which we evaluate on synthetic datasets. We apply PACP to a real dataset recorded in the Bedretto Underground Laboratory for Geosciences and Geoenergies, and discuss the observed improvements compared to the original baseline. Through our approach, we aim to address the pressing need for robust and accurate fault detection and characterization methods in seismology (Ouillon et al., 2008). Our synthetic catalog generator, the python implementation of PACP and Kamer et al.'s algorithm are available on GitHub[1]

2 Generating Synthetic Seismic Point-Clouds

To achieve the overarching goal of improving fault reconstruction algorithms, realistic and labelled seismic point cloud data is required to thoroughly and quantitatively

[1] `https://github.com/samhouliston/DSL_Quake_Fault_Reconstruction`

validate proposed modifications. Quake datasets measured from real events are readily available, however few are expertly-labelled and publicly available, and papers tend to verify their methods qualitatively based on how well they perform the clustering according to the (experts') naked eye, or using rather simple synthetic datasets. To validate our proposed method, we generate synthetic point clouds incorporating key fault features which we describe below.

2.1 Related work

Kamer et al. (2020) create synthetic test datasets by generating 20 randomly oriented fault planes with attributes in the ranges: strike angle $-90°$to $90°$, dip angle $45°$to $90°$, length 20 to 40 km, and width 5 to 15 km. These planes span a defined region, and each plane is sampled with varying point densities from 0.1 to 2 points per square kilometer in 15 steps, resulting in 609 to 14,475 points per set. They also introduce three background noise levels (at 5%, 10%, and 20%) creating 45 synthetic datasets. The resulting planes are mostly aligned, vary in densities, and cover a large domain, however, this dataset eludes key natural features found in faults which are particularly difficult to cluster.

Liu et al. (2021) generate synthetic microseismic events using an elaborate probabilistic scheme of defining a random gaussian mixture model and sampling points from it. The gaussian kernels represent individual faults with varying fracture scales and dip angles.

2.2 Main Fault Types

To form a dataset, we sample several different fault types that are realistic. We identify six fault types, that can be observed in real datasets, some of which are difficult to cluster.

Defining and generating a fault. We characterise a single fault as a plane, whose width is generally close to half its length. To generate a point cloud, we sample points on either side of the plane from a (normal) distribution such that the mean distance from the plane is slightly less than plane's width. In our implementation, we define a fault by its fault metadata: `(center, length, width, normal, length_axis, width_axis)`. We sample points at a fixed density of 150 points per m^2 of plane. This is done by calculating the number of points to sample `num_points`, then for each point, we sample a position on the plane: 'x-coordinates' uniformly between `[-width/2, width/2]`, and 'y-coordinates' uniformly between `[-length/2, length/2]`. Its distance from the plane is sampled from a Gaussian distribution of mean zero and standard deviation of `width/7`. Our more complex fault patterns are defined as a collection of simpler faults which we describe below.

Simple fault. The simple fault consists of a single plane. The plane hypocenter is randomly sampled in the space (at least `eps_center=0.5` distance from other centers), then the fault length axis, length, width axis, and width are sampled.

Bent fault. Faults don't always run perfectly straight - in fact, many will often curve or bend while still belonging to the same structure. For such a fault, our code generates a simple fault to which it appends another fault, joined at the edges of the planes. These connected planes are at an angle randomly generated between [-40, 40] degrees.

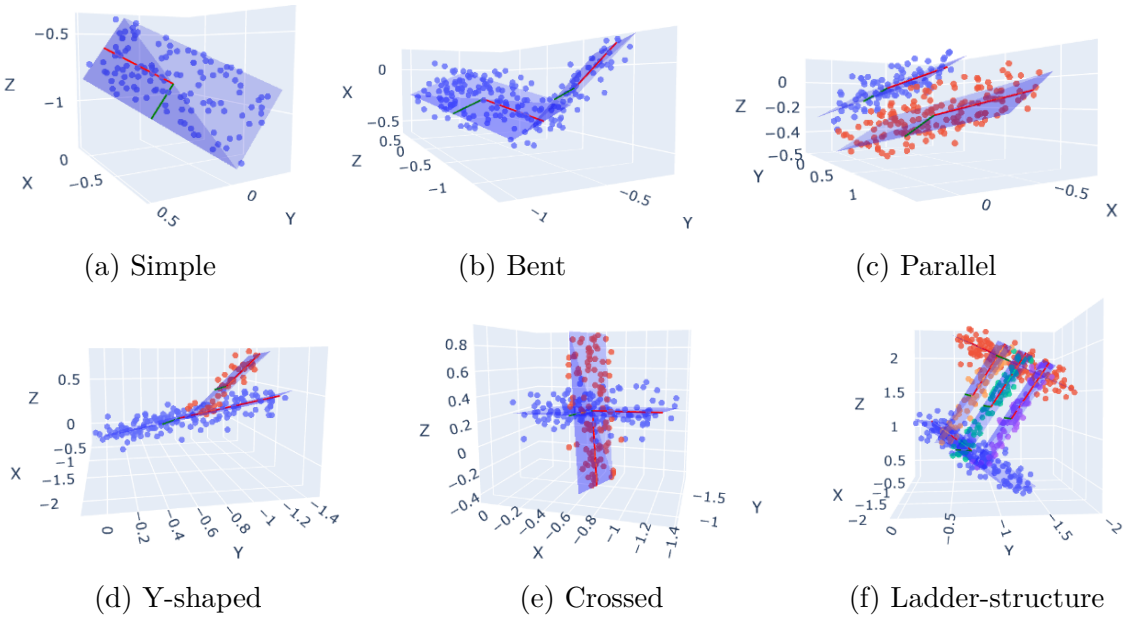
Parallel faults. Faults in a network are generally aligned along one of two main orthogonal directions. A common case that is sometimes difficult to cluster is when two separate faults are parallel and close. We generate this case by producing a simple fault, and generating a second fault with the same axes, but whose center is shifted away. The center shift is determined by the distance between the planes (which is sampled uniformly between `[0.5, 0.7]` times the first fault width) and the shift along the length axis (which is also sampled uniformly `[0.2, 0.7]` times the first fault length.

Y-shaped faults. Another tricky case is when a fault branches at one extremity - or when a smaller fault joins a larger one (see Figure 1. This fault is generated similarly to the bent fault, starting with a main simple fault and a secondary smaller fault. The secondary fault forms an angle sampled between $\pm$`[20, 35]` degrees with the main fault. Then the secondary width and length are sampled as 0.7 times the first width and 0.4 times the first length respectively. Then, the secondary center is computed such that faults just coincide at some point between the first fault's center and its extremity.

Crossed faults. Faults sometimes intersect at a pronounced angle. We form a case where two faults cross perpendicularly. This is formed by simple fault and a secondary fault with the same center and width axis but perpendicular length axes. The secondary length is the same but the width is set to 0.8 times the first fault's width.

Ladder-like structures. Real quake data sometimes exhibits a particular ladder-shaped structure (depicted in Figure 1). This ladder structure is characterized by two important parallel faults joined by numerous smaller faults, typically resembling echelons on a ladder as they are close to perpendicular to the main axis. Our code also generates such faults upon request, by first generating two (close to) parallel simple faults, and then a set of smaller parallel faults that lie close to the plane of

the original pair. In our implementation, we generate three equally spaced smaller faults perpendicular to the two main faults.



(a) Simple (b) Bent (c) Parallel

(d) Y-shaped (e) Crossed (f) Ladder-structure

Figure 1: Example realizations of basic fault types and structures

2.3 Dataset Generation

A dataset, or fault network, comprises of realizations of multiple basic fault types we outline above. Fault networks are generally aligned along two main perpendicular directions. To generate a dataset given the specific number of desired fault types, we start by sampling two perpendicular directions. Then when generating a fault, we sample a random center location while favouring either of the two axes. Additionally fault centers must be at least `eps_center=0.5` away from other centers in the dataset. For more detailed information about the implementation see Appendix 7.1.

Figure 2 depicts an example dataset of 2 bent faults, 1 cross structure, 1 Y-shaped fault, and 1 parallel structure. We notice the faults are aligned along two directions. When generating such faults, we record the 3D point cloud as well as the corresponding cluster label for each point.

Future improvements. The synthetic generation script could easily be improved by designing more specific functions to generate the metadata of different fault types and tuning the length, width and sampling distribution parameters. Additionally improving the scale adaptation of the domain would be good to generate larger datasets. Interesting data-driven ML methods for synthetic data are extremely interesting but may be fastidious to implement and simply unsuited to our problem,

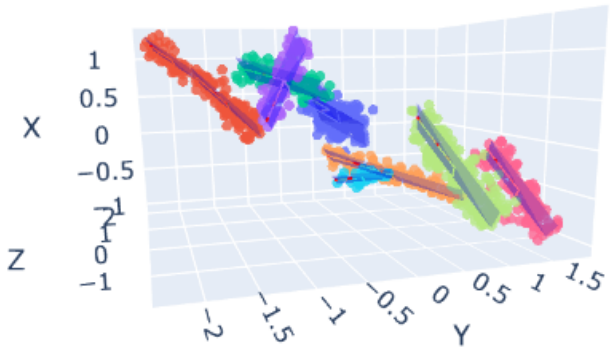


Figure 2: Example dataset

unless more real expert annotated data is available. More advanced models could aim to include the fractal nature of seismic events from the Gutenberg–Richter law.

3 Clustering Algorithms for Seismic Data

Problem statement. The overarching objective is to recover fault planes from a recorded seismic catalog, which has the form of a 3-dimensional (3D) point cloud. The idea is to first perform a clustering of the 3D point cloud, where each cluster should in theory model a fault, and then the fault plane can be extracted via principal component analysis of the cluster points (Ouillon et al., 2008). We investigate algorithms that perform the first step: extracting accurate clusters form the point cloud.

3.1 Related work

Since there is usually no ground truth available on the number of clusters in seismic catalogs, hierarchical clustering methods which construct a tree of partitions whose levels have increasing numbers of clusters are most suited. Hierarchical clustering methods can further be grouped into top-down, divisive approaches that iteratively split larger clusters into smaller ones, and bottom-up, agglomerative approaches that iteratively merge smaller clusters into larger ones (Saxena et al., 2017). The early notable methods tailored to seismic data are mostly divisive approaches, starting from a simple model and iteratively refining the clustering (Kamer et al., 2020). Ouillon et al. (2008) introduced OACD, a planar variant of the k-means algorithm, in which identified clusters are split and refined until their diameter is smaller than the global location error of the dataset. This was improved by Wang et al. (2013) with their ACLUD algorithm, which additionally consider changes in the location error between different clusters and accounts for uncertainty in the calculation of the distance of a data point to a given cluster. Palgunadi et al. (2020) augment

this approach in their method, g-ACLUD, by incorporating prior information on regional tectonic constraints, focal mechanisms and surface geological manifestation. A probabilistic approach to divisive hierarchical clustering is introduced by Ouillon and Sornette (2011), which represents clusters as 3D Gaussian kernels and iteratively split them, but choose the termination point of the splitting process and therefore the final number of clusters manually. Kamer et al. (2020) notice a common weakness of top-down methods: splits sometimes fail to converge to the local clusters as they may be attracted to the regions of high horizontal variance. Therefore, they introduce a bottom-up probabilistic method for seismic fault clustering, which our algorithm is based on. The method of Kamer et al. terminates the agglomeration process based on Bayesian Information Criterion and thus does not require a manual selection of the final number of clusters.

3.2 Starting Point: Probabilistic Agglomerative Clustering

Kamer et al. (2020) present an approach to agglomerative clustering tailored specifically to seismic fault network reconstruction, which we outline in black in Algorithm 1. It takes as input a 3-dimensional (3D) point cloud, where each data point corresponds to the geographic location of a seismic event. The algorithm comprises 2 phases, the Initialization phase and the Probabilistic Agglomerative Merging phase and can be regarded as a variant of a Gaussian Mixture Model.

Initialization The purpose of this phase is to define the starting clusters for the algorithm. Since the performance of the second phase heavily depends on the quality of the initialization, they are defined based on agglomerative hierarchical clustering instead of using a randomized procedure. The phase requires the min_cluster_size as a hyperparameter, which defines the number of assigned points from which on a given cluster is considered valid. A complete agglomerative tree of the dataset is formed using Ward's linkage. This tree is then traversed from root to leaves while counting the number of valid clusters (cluster size $\geq$ min_cluster_size) at every level. The cluster assignment defined by the level with the largest number of valid clusters is used as initial cluster assignment, the so-called capacity clusters of the dataset. Next, the mixture model is initialized based on the capacity clusters. A 3D-Gaussian kernel is then fit to the points of every valid capacity cluster in the cluster assignment by determining their sample mean μ and covariance Σ as well as the weight w of the cluster. A uniform background kernel is fit to all remaining points from invalid capacity clusters. Additionally, every kernel is equipped with a minimum volume rectangular bounding box which encloses the maximum of all points in the corresponding capacity cluster and the space within $\sqrt{12}$ standard deviations of the kernel's mean. The cluster assignment is then recomputed using maximum likelihood estimation over all kernels for each data point, the weights and bounding boxes are adjusted afterwards based on the new assignment.

Probabilistic Agglomerative Merging During this phase, the Gaussian kernels in the mixture model are merged iteratively. At every iteration, a list of candidate pairs is determined. This list entails all pairs of kernels that have not been ruled out as candidates during an earlier iteration and whose bounding boxes overlap. For each pair (a, b), the kernel ab with mean $\mu_{ab} = \frac{w_a}{w_a+w_b}\mu_a + \frac{w_b}{w_a+w_b}\mu_b$, covariance $\Sigma_{ab} = \frac{w_a}{w_a+w_b}(\Sigma_a + (\mu_a - \mu_{ab})(\mu_a - \mu_{ab})^T) + \frac{w_b}{w_a+w_b}(\Sigma_b + (\mu_b - \mu_{ab})(\mu_b - \mu_{ab})^T)$ and weight $w_{ab} = w_a + w_b$ that results from merging the two individual members is computed. The bounding box of the merged kernel is the bounding box that encloses all 16 corner points of the bounding boxes of a and b. Then, the information gain G_{BIC} from merging the two kernels is calculated, which is given by the difference in Bayesian Information Criterion (BIC) before and after merging the two kernels:

$$G_{\text{BIC}}(a, b, C_K) = BIC(C_K) - BIC(C_K \setminus \{a, b\} \cup ab). \tag{1}$$

BIC trades data likelihood with model complexity (Schwarz, 1978) and is defined as

$$BIC(C_K) = -\sum_{i}^{N} \log\left(\sum_{j}^{K} p(x_i|\mu_j, \Sigma_j, w_j, B_j)\right) + \frac{10K - 1}{2}\log(N),$$

where C_K denotes the kernel configuration, K the number of kernels and N the number of data points. The likelihood of point i under kernel j subject to the condition that i lies within the kernel's bounding box B_j is denoted by $p(x_i|\mu_j, \Sigma_j, w_j, B_j)$. If the information gain is negative, the kernel pair is ruled out and not further considered for merging. For all remaining candidate pairs with positive information gain, the pair (a, b) with highest gain is merged. All other candidate pairs (c, d) whose individual members form a different candidate pair (e.g. (a, c)) with either a or b or directly contain a or b are removed from the list. This procedure is repeated until no candidate pairs with positive information gain are left. The entire merging phase is repeated until no more initial candidate pairs can be identified.

3.3 Favoring Cluster Planarity

The method described by Kamer et al. does not include any model assumptions related to the properties of seismic data. Since the Gaussian cluster model is quite flexible, it is still possible in theory to capture the intricacies of the clusters in seismic point clouds. However, on some typical patterns found in seismic data, we observed that the algorithm tends towards an overly simple cluster structure as a result of the trade-off between likelihood increase and model complexity in BIC. This becomes especially evident in ladder-like structures, where the echelons, the near-perpendicular connections between the two main faults, typically consist of a small number of data points compared to main faults. Thus, the complexity penalty in BIC favors merging of the echelons into the clusters representing the main faults.

Algorithm 1 Probabilistic Agglomerative Clustering Favoring Planarity (PACP)

Input: Point-cloud dataset of measurements.
Output: Gaussian kernels that represent individual fault clusters.
I. Initialization

- Form agglomeration tree of clusters based on Ward's criterion
- Pick level with most clusters of size bigger than `min_cluster_size`
- Initialize mixture model:
  1. Fit Gaussian kernel to valid "capacity" clusters ($\geq$ `min_cluster_size`)
  2. Fit uniform background kernel to remaining points
- Fit one bounding box per cluster/kernel

II. Probabilistic Agglomerative Merging
while unconverged (Gain > 0):

for every pair of clusters with overlapping bounding boxes:

- Compute merged kernel
- Compute Gain from merging

end for

while pairs with Gain > 0 remaining:

- Merge pair (A,B) with highest Gain
- Remove all other pairs containing A or B
- Remove pairs (C,D) having positive gain with D or C (e.g.: Gain(A,C)>0)

end while

- Re-assign points to kernels with maximum likelihood
- Re-fit mixture model of Gaussian + background kernels.

end while
Return: Gaussian kernels representing fault segments

To combat this, we propose the introduction of a penalty term into the merging criterion which penalizes merging of two candidate clusters that are not well aligned. The justification for this lies in the natural structure of the true clusters in seismic data: Since they are planar, we assume that the smaller clusters in the beginning of the iterative merging stage are also planar. In extension, two small planar clusters that represent different parts of the same true structure are parallel while small planar clusters that represent parts of two different true structures (e.g. a main

fault and an echelon in a ladder-like structure) are oriented at an angle to each other. One can measure the relative orientation (or alignment) of two clusters a and b as the absolute cosine similarity between their normals. The normal $\vec{n}_i$ of a cluster i can be determined as the normalized eigenvector corresponding to the smallest principal component of the data points assigned to i. This results in the following augmentation of the merging criterion detailed in Equation 1 for evaluating candidate kernels a and b:

$$G_{\text{align}}(a, b, C_K) = G_{\text{BIC}}(a, b, C_K) - \lambda(1 - |\langle \vec{n}_a, \vec{n}_b \rangle|). \tag{2}$$

λ denotes here an additional hyperparameter that scales the alignment term and needs to be tuned on every dataset. Imposing a penalty with respect to alignment has the added effect that the resulting clusters are more likely to be planar, since merging two misaligned planar clusters into a more isotropic cluster is discouraged. With this modification of the algorithm, the min_cluster_size must not be too small, since the initial clusters must already accurately represent the normals of the larger structures they are part of. In practice, we found that min_cluster_size $\in \{10, 15\}$ works well. Note that in contrast to Kamer et al., we reassign every data point to the Gaussian kernel under which it has maximum likelihood and consequently refit all kernel parameters based on this new assignment at the end of every iteration in the Probabilistic Agglomerative Merging phase (highlighted in red in Algorithm 1). The reason for this is that we do not have any information about the true data generating process, and therefore cannot assume that the average kernel computed from the two separate kernels in a merged pair is the best representation of the newly formed cluster.

3.4 Metrics

To measure the quality of a clustering with respect to a ground truth clustering, we use the Adjusted Rand Index (ARI) which is a measure of the similarity between two data clusterings. It is a correction of the Rand Index, which is a basic measure of similarity between two clusterings, but has the disadvantage of being sensitive to chance (Hubert and Arabie, 1985). The ARI ranges between $[-1, 1]$ with 1 indicating a perfect agreement between the clusterings, an ARI of 0 indicating that the clustering is no better than random assignment and negative values indicating less agreement than expected by chance.

4 Experimental Results

4.1 Evaluation on Synthetic Data

We compare our method, PACP, to the algorithm of Kamer et al. and standard K-Means clustering on 20 realizations of a synthetic mixed test dataset comprising of

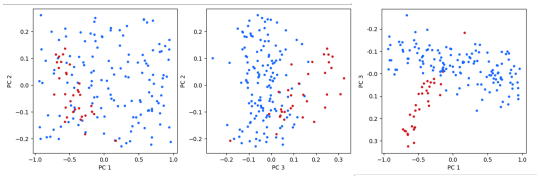
11 faults (1 parallel (2 faults), 1 Y-shaped (2 faults), 1 cross (2 faults) and 1 ladder structure (5 faults)). We also evaluate the methods on 20 realizations each of an individual Y-fault, cross and ladder structure. Based on a visual evaluation of the performance, we chose min_cluster_size= 10 for all datasets, and $\lambda = 15, 10, 10, 20$ for mixed dataset, Y-fault, cross and ladder. Table 1 summarises the mean ARI performance over all realizations.

Performance Comparison			
	PACP	Kamer et al.	K-Means
Y-fault	**0.723**	0.616	0.076
Cross	**0.775**	0.673	0.365
Ladder	**0.863**	0.736	0.051
Mixed Dataset	**0.810**	0.728	0.495

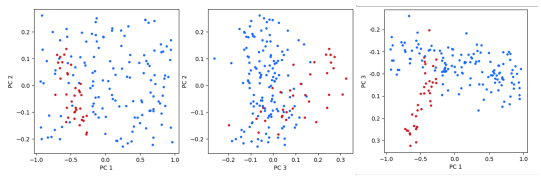
Table 1: Adjusted Rand Index over 20 trials, mixed dataset of 11 faults

PACP surpasses the original algorithm and standard K-Means in terms of ARI on all synthetic datasets. We observe the standard K-Means performs poorly compared to the seismicity-tailored methods. We also notice that for larger datasets (i.e., ladder and mixed dataset), our method and Kamer et al.'s have higher ARI in general. This is due to the ARI being insensitive to small differences in clustering when dealing with many points. Since the relevant structures such as the echelons of the ladder comprise fewer points than the main directions of the ladder, them being incorrectly partitioned does not have a large effect on the ARI. The phenomenon can also be observed in a qualitative assessment of the clusterings.

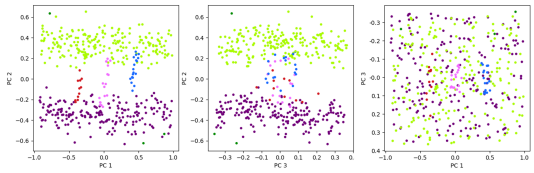
Qualitative Assessment. Figure 3 provides a visual comparison of the clustering obtained with PACP and with Kamer et al.'s method for a single realization of a Y-fault and a ladder structure. We observe in the ladder structure that our method (Figure 3(c) PC2 vs PC1) captures the smaller echelons, while Kamer et al.'s (Figure 3(d) PC2 vs PC1) merges them into the clusters representing the two main faults. In the Y-shape, we can observe that Kamer et al.'s method (Figure 3(b) PC3 vs PC1) assigns some points to the red cluster which are oriented at an angle to the remaining points in that cluster. This is prevented in PACP (Figure 3 (a) PC3 vs PC1) due to the alignment constraint.



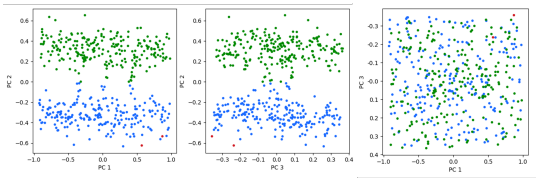
(a) PACP, Y-shape



(b) Kamer et al., Y-shape



(c) PACP, ladder structure



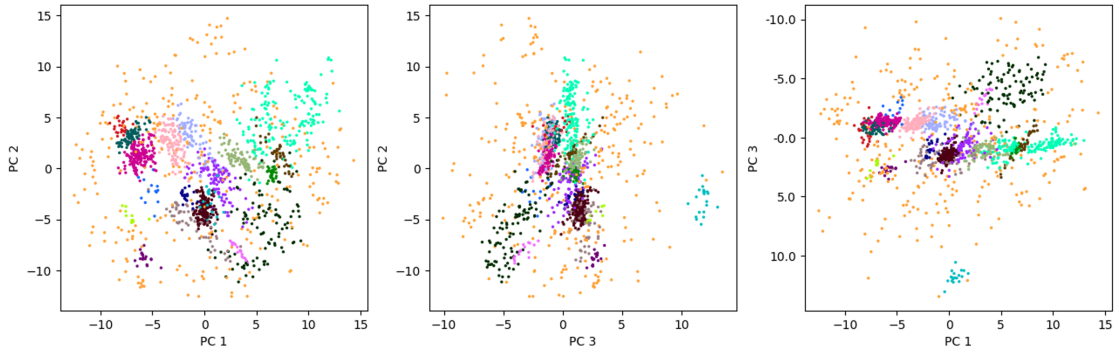
(d) Kamer et al., ladder structure

Figure 3: Example visulalisations of clustering on synthetic data. 3D plots are projected along principal components PC1, PC2, PC3 of the data.

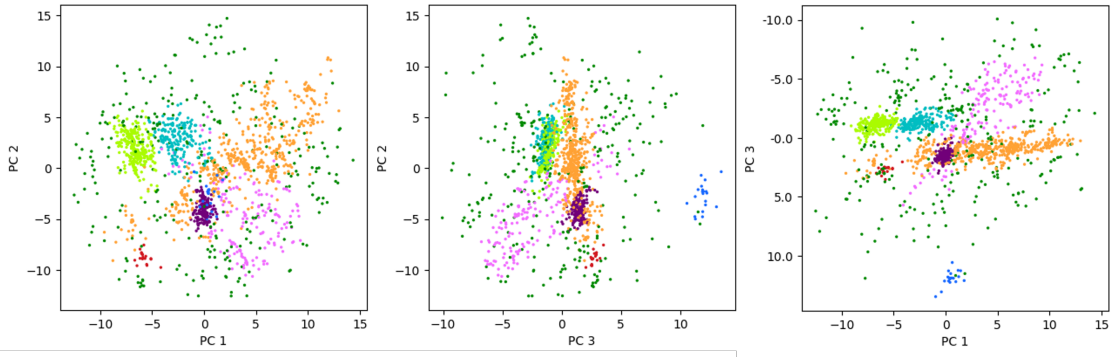
4.2 Evaluation on the Bedretto Dataset

The Bedretto Dataset is a high-resolution dataset of 2723 points displaying one large ladder-like structure. The measurements were collected over 48 hours from a deliberately induced quake in the Bedretto Underground Laboratory for Geosciences and Geoenergies. Our experiments use the first 15 hours of recorded data points.

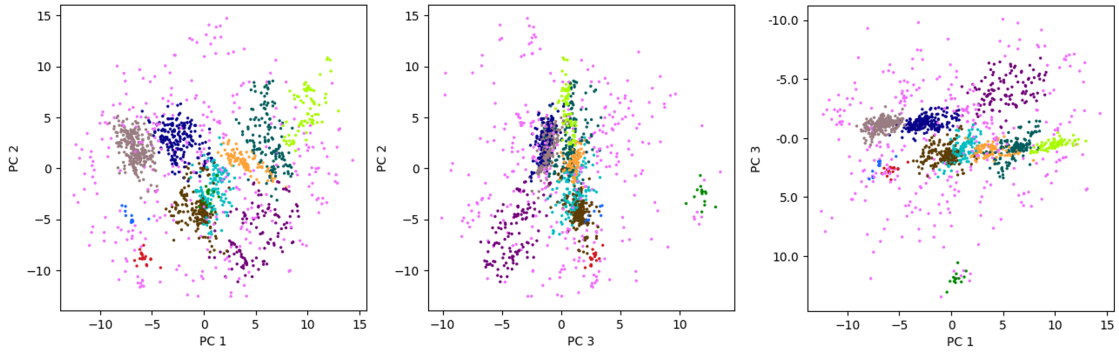
We present our result obtained with the following hyperparameters: For PACP, we used penalty scale $\lambda = 20$, and min_cluster_size=10, for Kamer et al.'s method, we used min_cluster_size=10,15. Since there is no ground truth available on for this dataset, we have to resort to a qualitative analysis of the results. Figure 4 showcases the clusters obtained with PACP and Kamer et al.'s method respectively. The ladder-like structure can be recognized best in the PC3 vs PC1 visualizations. We did not perform K-Means clustering on this dataset since the unknown true number of clusters is a necessary hyperparameter of the algorithm. We observe that PACP is able to identify some of the echelons of the ladder structure (Figure 4(a) PC3 vs PC1 plot, brown, purple, navy cluster), but also fragments the two main faults into multiple clusters. Using Kamer et al.'s method with min_cluster_size=15, we find that it is able to identify the two main faults, but similarly to our observations on the synthetic ladder dataset does not recognize the perpendicular echelons (Figure 4(b) PC3 vs PC1). In contrast, when using min_cluster_size=10, the resulting clustering is more fragmented and looks more similar to the results obtained with PACP (Figure 4(c) PC3 vs PC1). Upon closer inspection, it can be observed that while this clustering is more fragmented, it does not sharply capture any of the echelons like PACP but merely fragments the main fault directions. This change between the different parameter configurations also highlights that the algorithm is sensitive to hyperparameters.



(a) PACP (min_sz_cluster=10, $\lambda = 20$)



(b) Kamer et al. (min_sz_cluster=15)



(c) Kamer et al. (min_sz_cluster=10)

Figure 4: Visualization of clusters found on the Bedretto dataset (first 15 hours). 3D plots are projected along principal components PC1, PC2, PC3 of the data.

5 Discussion

We observed that PACP is able to outperform both the baseline of Kamer et al. as well as the naive baseline of K-Means in terms of Adjusted Rand Index on synthetic datasets. On the real Bedretto dataset, the result is not as conclusive. While

PACP better captures the fine structures of the ladder, Kamer et al.'s method correctly identifies the two main fault directions when tuned appropriately. Thus, a combination of both methods to extract the main and fine structures separately would be an interesting direction for future investigations.

On the practical front, both PACP and Kamer et al.'s method are very computationally intensive and slow to run. One improvement would be to parallelize the Probabilistic Agglomerative Merging phase by first partitioning the dataset. When doing so, it has to be ensured that complicated structures (e.g. a ladder) are not split. Ideally, each complex structure would end up in a separate partition. Such a procedure would provide the added benefit of being able to tune hyperparameters separately for each partition, which could improve clustering performance. To do this efficiently, a method to automatically determine the optimal value of λ is needed.

Our algorithm is majorly probabilistic which has the advantage of being robust to noise and low-probability samples and of converging automatically (not requiring manual termination), however it does not fully exploit known prior information of the data: that it is generally aligned along two directions and specific structures often recur. Perhaps combining it with a RANSAC-like algorithm that explicitly fits planes would be beneficial.

6 Conclusion

In conclusion, our contributions are the following: we provide a synthetic dataset that generates realistic data, we re-implement Kamer et al.'s algorithm in python for easier comparison and possible future integration with ML methods, and lastly we propose PACP, which introduces an additional alignment penalty to favour planarity and a re-fitting stage to increase the model accuracy. Our extensions to the algorithm show quantitative improvements on the synthetic data and qualitatively seem to capture perpendicular and finer fault structures better on the Bedretto dataset. We also advocate for a wider comparison of clustering methods on seismic data and propose some avenues of investigation to improve our method.

References

Yavor Kamer, Guy Ouillon, and Didier Sornette. Fault network reconstruction using agglomerative clustering: applications to southern californian seismicity. *Natural Hazards and Earth System Sciences*, 20(12):3611–3625, 2020. doi: 10.5194/nhess-20-3611-2020.

Andreas Plesch, John H Shaw, Christine Benson, William A Bryant, Sara Carena, Michele Cooke, James Dolan, Gary Fuis, Eldon Gath, Lisa Grant, et al. Community

fault model (cfm) for southern california. *Bulletin of the Seismological Society of America*, 97(6):1793–1802, 2007.

Guy Ouillon and Didier Sornette. Segmentation of fault networks determined from spatial clustering of earthquakes. *Journal of Geophysical Research*, 116(B2), 2011. doi: 10.1029/2010jb007752.

Mitiyasu Ohnaka. *The Physics of Rock Failure and Earthquakes*. Cambridge University Press, 2013. doi: 10.1017/CBO9781139342865.

Qingkai Kong, Daniel T. Trugman, Zachary E. Ross, Michael J. Bianco, Brendan J. Meade, and Peter Gerstoft. Machine Learning in Seismology: Turning Data into Insights. *Seismological Research Letters*, 90(1):3–14, 2018. doi: 10.1785/0220180259.

Guy Ouillon, Caroline Ducorbier, and Didier Sornette. Automatic reconstruction of fault networks from seismicity catalogs: Three-dimensional optimal anisotropic dynamic clustering. *Journal of Geophysical Research (Solid Earth)*, 113(B1): B01306, 2008. doi: 10.1029/2007JB005032.

Xing Liu, Yan Jin, Botao Lin, Qixing Zhang, and Shiming Wei. An integrated 3d fracture network reconstruction method based on microseismic events. *Journal of Natural Gas Science and Engineering*, 95:104182, 2021. doi: 10.1016/j.jngse.2021.104182.

Amit Saxena, Mukesh Prasad, Akshansh Gupta, Neha Bharill, Om Prakash Patel, Aruna Tiwari, Meng Joo Er, Weiping Ding, and Chin-Teng Lin. A review of clustering techniques and developments. *Neurocomputing*, 267:664–681, 2017. doi: 10.1016/j.neucom.2017.06.053.

Yaming Wang, Guy Ouillon, Jochen Woessner, Didier Sornette, and Stephan Husen. Automatic reconstruction of fault networks from seismicity catalogs including location uncertainty. *Journal of Geophysical Research: Solid Earth*, 118(11): 5956–5975, 2013. doi: 10.1002/2013JB010164.

Kadek Hendrawan Palgunadi, Alice-Agnes Gabriel, Thomas Ulrich, José Ángel López-Comino, and Paul Martin Mai. Dynamic Fault Interaction during a Fluid-Injection-Induced Earthquake: The 2017 Mw 5.5 Pohang Event. *Bulletin of the Seismological Society of America*, 110(5):2328–2349, 2020. doi: 10.1785/0120200106.

Gideon Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6 (2):461 – 464, 1978. doi: 10.1214/aos/1176344136.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985. doi: 10.1007/BF01908075.

7 Appendix

7.1 Synthetic Data Generator Implementation Details

Using the script. A dataset of a fault network comprising of a point-cloud and corresponding cluster labels is generated by our script called `generate_synthetic_dataset.py` and can be executed via the command line using the following syntax:

```
python generate_synthetic_dataset.py <arguments>
```

The script accepts the following arguments specifying the parameters for the dataset generation:

`-o`, `-output`: Path to the output file.

`-v`, `-verbose`: Enable verbose mode.

`-visualise`: Visualise generated dataset.

`-n_simple_faults`: Number of simple faults. Default value: 1.

`-n_bent_faults`: Number of bent faults. Default value: 0.

`-n_cross_faults`: Number of cross faults. Default value: 0.

`-n_Y_faults`: Number of Y faults. Default value: 0.

`-n_parallel_faults`: Number of parallel faults. Default value: 0.

`-n_structures`: Number of structures. Default value: 0.

Generation workflow. We present the workflow and functions hierarchically:

1. Given the input arguments, the script performs:

```
fault_points_list, labels = generate_dataset(
                n_simple_faults,
                n_bent_faults,
                n_cross_faults,
                n_Y_faults,
                n_parallel_faults,
                n_structures,
                VERBOSE)
```

2. The `generate_dataset` function first calls the `generate_faults_metadata` function to obtain a list containing the metadata of each desired fault. This is where the parameters `eps_center` and `domain_scale` are determined.

```
fault_metadata_list = generate_faults_metadata(
                        n_simple_faults,
                        n_bent_faults,
                        n_cross_faults,
                        n_Y_faults,
                        n_parallel_faults,
                        n_structures,
                        eps_center=0.5,
                        domain_scale=1,
                        VERBOSE=False)
```

3. Once the metadata list is obtained, we sample a point cloud for every fault using the `generate_point_list_from_metadata_list` function. This function will iterate over the list of metadata and keep track of the type of fault to generate thanks to a specific ordering and known indices/numbers of faults to generate. Each fault type consists of one or multiple simple planes, this function appropriately calls the `generate_points_from_plane` for every plane.

```
fault_points_list, labels = generate_point_list_from_metadata_list(
                      fault_metadata_list,
                      n_simple_faults,
                      n_bent_faults,
                      n_cross_faults,
                      n_Y_faults,
                      n_parallel_faults)
```

4. It is inside the `generate_points_from_plane` function that the point sampling density parameter `density` is fixed to 150.

```
generate_points_from_plane(
                      center,
                      length,
                      width,
                      normal,
                      length_axis,
                     width_axis, distribution_mode='normal',
```

```
domain_scale=1)
```