

CMPSC 463

Project 1: Time-Series Clustering and Segment Analysis

on PulseDB Using Divide-and-Conquer Algorithms

Erand Vejseli

* **Description of project**

The purpose of this project is to design and implement a system that performs unsupervised clustering of biomedical time-series signals using algorithmic reasoning instead of machine learning heuristics. The data used in this study comes from the PulseDB dataset, which contains physiological signals such as ECG, PPG, and ABP (Arterial Blood Pressure).

The main goal of this project is to group similar 10-second time-series segments based on their waveform similarity and analyze their internal patterns. The system uses three key algorithms:

1. A Divide-and-Conquer clustering algorithm that recursively partitions the dataset into smaller groups based on correlation similarity.
2. A Closest Pair algorithm that identifies the two most similar signals within each cluster to validate cluster cohesion.
3. Kadane’s Algorithm that detects the most active or anomalous interval within each individual signal segment.

This project emphasizes algorithm design principles such as recursion, partitioning, and computational reasoning. It demonstrates that meaningful time-series segmentation can be achieved without relying on machine learning or external libraries.

* **Installation and usage**

The program was written in Python. I personally use PyCharm as my IDE. The required packages to run this program are numpy, h5py and matplotlib. You can install all those by running these commands in your Python terminal:

pip install numpy

pip install h5py

pip install matplotlib

The project uses VitalDB\_AAMI\_Test\_Subset.mat, which contains multiple time-series signals from PulseDB. The code automatically reads 1000 segments from one channel (ABP) and trims each segment to 800 samples to represent approximately 10 seconds of data.

To run this program, simply open the main.py file in your IDE,after making sure all the libraires above are already installed, and simply hit run. Make sure the .mat file is placed in the same folder as main.py. A small message will appear in your IDE letting you known that the data was processed, and the real output will be saved in a folder named out, which will be created if not already existing.

* **Structure of Code**

Start Program

Load the dataset via h5py

Preprocess the data:

Select ABP channel, limit to 1000 signals, and normalize (Using the z-score)

Divide & Conquer Clustering

Finds the farthest pair, splits recursively and stops at threshold

Closest Pair Algorithm

Compute correlations and finds most similar pair per cluster

Kadane’s Algorithm

Detects most active interval per segment and highlight in plot

Generate Outputs

clusters.csv, closest\_pairs.csv kadane\_windows.csv, example plots (.png)

End program

The overall structure of the program follows a divide-and-conquer workflow, where each part of the system handles one key step in the analysis.  
The block diagram above shows the data flow from loading the PulseDB dataset to generating outputs and plots. Below is an explanation of how the code is organized and what each part does.

All of the logic is implemented in a single Python file, main.py, which is organized into modular sections:

* Configuration Section – defines constants such as the file path, number of samples, and output directory.
* Data Loading Function (load\_signals\_from\_mat) – reads the VitalDB .mat file using the h5py library and extracts the arterial blood pressure (ABP) channel for analysis.
* Helper Functions (zscore\_rows, corr\_distance, farthest\_pair) – perform normalization and compute correlation-based similarity between time-series segments.
  + The z-score normalization step transforms each signal so it has a mean of 0 and a standard deviation of 1. This is the same concept used in statistics to make values comparable on the same scale(I remember studying this concept deeply in my STAT 318 course). It ensures that signals with higher amplitudes do not dominate the similarity calculation.
  + The correlation distance function measures how similar two signals are in shape rather than in raw amplitude. A smaller distance means the two signals have similar patterns, even if their actual values are different. This approach is important for clustering physiological data, since the signals may differ in range but still follow the same trend.
* Divide-and-Conquer Clustering (dnc\_cluster) – recursively splits the dataset into smaller, more homogeneous clusters based on correlation distance.
* Closest Pair Function (closest\_pair\_in\_cluster) – finds the most similar pair of signals within each cluster using a brute-force comparison.
* Kadane’s Algorithm (kadane\_interval\_on\_diff) – detects the most active or anomalous interval within each signal segment.
* Saving and Plotting Utilities – export results (CSV files and plots) for clusters, pairs, and Kadane intervals.
* Main Execution Block – runs the complete pipeline from data loading to final output generation.

Although implemented in a single file, the code is divided into clear, reusable functions. Each function performs one specific task and can be tested independently, making the structure modular and easy to maintain. The use of helper functions (like corr\_distance and zscore\_rows) prevents code repetition and simplifies debugging.

* **Description of algorithms**

**Divide-and-Conquer Clustering**

This algorithm follows a recursive divide and conquer approach to group similar time-series segments. At each step:

1. Computes correlation distances among signals in the current group.
2. Identifies the two farthest signals (most dissimilar pair).
3. Splits the group into two smaller clusters based on which signal each element is more similar to.
4. Continues recursively until the clusters reach a minimum size or a maximum recursion depth.

This approach efficiently builds a binary tree of clusters.

**Closest Pair Algorithm**

Within each cluster, this step finds the two most similar time-series segments using a brute-force pairwise comparison. Although the divide-and-conquer version of the closest pair algorithm exists and we studied it in this class, the brute-force version is simpler and more efficient here due to the small cluster sizes. It validates cluster quality by ensuring that the pair with the smallest correlation distance truly belongs together.

**Kadane’s Algorithm for Active Interval Detection**

Kadane’s algorithm finds the maximum subarray sum in a sequence. In this project, it is applied to the first difference of each signal, which represents local changes over time.  
By locating the segment with the highest cumulative positive change, the algorithm highlights the most active or significant interval of the waveform.

* **Verification of the functionality with toy example**

Before using the full dataset of 1,000 signals, I tested each algorithm on very small and simple examples to make sure they work correctly.  
This helped confirm that the clustering, closest-pair, and Kadane’s algorithm were all giving the right kind of results before running the complete program.

**5.1 Divide-and-Conquer Clustering (Toy Test)**

To test the clustering function, I made a few short fake signals, some with smooth up-and-down wave shapes and others with sharp jumps. When I ran the dnc\_cluster() function, it correctly separated them into two groups: one for smooth signals and one for the sharp ones. This shows that the clustering logic can correctly split signals based on how similar their shapes are.

**5.2 Closest Pair Algorithm (Toy Test)**

Next, I tested the closest-pair function with three short lists:

x1 = [1, 2, 3, 4, 5]

x2 = [2, 3, 4, 5, 6]

x3 = [10, 9, 8, 7, 6]

The function compared all of them and correctly found that x1 and x2 were the most similar pair. That makes sense since they both increase steadily, while x3 goes in the opposite direction. This confirms that the function for comparing signals by correlation is working right.

**Kadane’s Algorithm (Toy Test)**

Finally, I tested Kadane’s algorithm with a small example list:

x = [-2, 1, -3, 4, -1, 2, 1, -5, 4]

The algorithm found the section [4, -1, 2, 1] as the part with the largest total sum, which equals 6. This is the correct answer and shows that the algorithm can detect the most active interval in a signal.

* **Execution results with 1000 time series**

After running the program on 1,000 ABP segments from the PulseDB dataset, the system generated several outputs saved in the /out directory.  
These include:

* Cluster assignment files (clusters.csv)
* Closest pair information (closest\_pairs.csv)
* Kadane intervals (kadane\_windows.csv)
* Visualizations of representative and closest signals for each cluster

Example Visualizations:

This plot shows the representative signal of Cluster 0, along with the most active interval (highlighted region) detected by Kadane’s algorithm.

*A screenshot of a computer

AI-generated content may be incorrect.*

This plot illustrates the closest pair within Cluster 0.  
The distance value (d=0.0354) represents the correlation distance. Smaller values indicate higher similarity.

*A screenshot of a computer

AI-generated content may be incorrect.*

This plot shows another cluster example, where the periodic peaks reflect repeating ABP waveforms.  
The shaded blue area marks the segment of maximum activity, identified through Kadane’s algorithm.

*A screenshot of a computer

AI-generated content may be incorrect.*

* **Discussion on execution results**

This project demonstrates that algorithmic design methods such as divide-and-conquer can effectively analyze biomedical time-series data. Without relying on machine learning libraries, the implemented approach achieved meaningful grouping and activity detection purely through recursive logic and mathematical operations.

**Insights:**

* The correlation-based clustering worked very well for ABP signals since they share similar periodic wave patterns, especially when normalized with z-scores.
* The closest-pair function helped confirm the cohesion of clusters by showing which two signals were most alike within each group.
* Kadane’s algorithm added interpretability by clearly identifying the most active or eventful intervals in each signal segment.

**Challenges:**

Developing this project was not easy and involved several learning steps and debugging challenges:

* Understanding and using NumPy for the first time was one of the biggest challenges.  
  I had to learn how NumPy handles arrays, axes, and vectorized operations, especially for mean, standard deviation, and correlation calculations.  
  Small things like axis=1 or keepdims=True were very confusing at first but later made sense when I understood how they keep array dimensions consistent. I had to use the help of AI tools too quite a few times. I can say that I am more familiar with it now, rather than I learned it.
* Working with the .mat dataset was another difficulty.  
  The PulseDB .mat file did not have a simple structure, it contained nested groups inside the HDF5 format. And I could not open the file right away in Microsoft Access too, which just made the whole situation much more complicated.  
  Finding the right data path (f["VitalDB"]["Signals"]) using the h5py library took a lot of time.
* Debugging shape mismatches (for example, when signals didn’t have the same dimensions or channel count) required careful handling. I learned to add conditions to reshape arrays when necessary.
* Performance issues also appeared.  
  The closest-pair search uses brute-force comparisons, and for very large clusters this becomes slow. I had to rely on the divide-and-conquer clustering to keep each cluster small enough to make it manageable.
* Interpreting the results visually was also a learning process.  
  Understanding how the shaded Kadane regions relate to physical activity in the ABP waveform required going back and forth between the plots and algorithm logic.

**Limitations and Future Improvements**

* Implementing a true divide-and-conquer version of the closest-pair algorithm could improve performance and scalability for larger datasets.
* The program currently focuses on a single signal type (ABP).  
  Expanding it to analyze multiple physiological channels (such as ECG and PPG) at the same time could reveal more relationships between signals.
* **Conclusions**

Overall, this project achieved the main goals by combining three algorithmic techniques: divide-and-conquer clustering, closest-pair search, and Kadane’s maximum subarray analysis, to process 1,000 PulseDB time-series segments. The results show that even without machine learning, traditional algorithmic thinking can successfully organize and interpret biomedical signal data.

This project strengthened my skills in algorithm design, NumPy programming, and scientific visualization.