BÀI BÁO CÁO: Project Cuối Kì
XỬ LÝ ĐA CHIỀU

19110311 - Nguyễn Ngô Trung Hậu
19110315 - Trịnh Ngọc Hiến

Time Series Clustering

1 Giai đoạn tiền xử lý:

- Trước tiên, chúng ta xử lý các giá trị thô ban đầu có được từ Dataset. Bằng việc đánh số tên các file thành 1 kiểu nhất định (VD: 1.txt) để dễ dàng truy xuất trong Google Colab. Tổng cộng có tất cả 979 file Time-Series Dataset.

- Tiến hành đưa dữ liệu đã được đánh số lên Google Colab để xử lý dữ liệu thô.

Hàm truy xuất file:

```
[2]  1 def truyxuat_file(i):
     2   t = open ( str(i)+'.txt' , 'r')
     3   l = []
     4   h = []
     5   l = [ line.split() for line in t]
     6   x = np.array(l)
     7   for i in x:
     8     h.append(i.astype(float))
     9   h = np.array(h)
    10   return h
    11
```

Hình 1: Hàm truy xuất file giúp hỗ trợ việc đọc 979 file Time-Series.

- Bước tiếp theo, ta lấy tất cả độ dài của các file và lưu vào 1 biến. Sau đó ta tiến hành tìm file có độ dài lớn nhất. Từ đó, ta chọn độ dài phù hợp để biến đổi Fourier các file về cùng 1 độ dài.

- Tiến hành chia nhỏ các dữ liệu của từng file thành 3 cột dữ liệu tương ứng với 3 cột trong 1 file vào 1 biến để thuận tiện cho việc biến đổi Fourier dữ liệu của các file.

- Tiến hành tạo biến để lưu hiệu các độ dài của các file so với độ dài mong muốn nhằm hỗ trợ cho việc thực hiện biến đổi Fourier.

- Tiến hành sử dụng hàm ngoại vi Fourier, để biến đổi Fourier các dữ liệu có được về cùng 1 độ dài.

Hàm ngoại suy Fourier:

```
[11]  1 from numpy import fft
      2
      3 def fourierExtrapolation(x, n_predict):
      4     n = x.size
      5     n_harm = 10                     # number of harmonics in model
      6     t = np.arange(0, n)
      7     p = np.polyfit(t, x, 1)         # find linear trend in x
      8     x_notrend = x - p[0] * t        # detrended x
      9     x_freqdom = fft.fft(x_notrend)  # detrended x in frequency domain
     10     f = fft.fftfreq(n)              # frequencies
     11     indexes = list(range(n))
     12     # sort indexes by frequency, lower -> higher
     13     indexes.sort(key = lambda i: np.absolute(f[i]))
     14
     15     t = np.arange(0, n + n_predict)
     16     restored_sig = np.zeros(t.size)
     17     for i in indexes[:1 + n_harm * 2]:
     18         ampli = np.absolute(x_freqdom[i]) / n   # amplitude
     19         phase = np.angle(x_freqdom[i])          # phase
     20         restored_sig += ampli * np.cos(2 * np.pi * f[i] * t + phase)
     21     return restored_sig + p[0] * t
```

Hình 2: Hàm ngoại suy Fourier dùng để biến đổi Fourier.

- Về phần hàm biến đổi Fourier:

+ Biến đổi Fourier nhanh (FFT) là một thuật toán rất hiệu quả để tính toán Biến đổi Fourier rời rạc (DFT)

+ Phép Biến đổi Fourier rời rạc (DFT) được định nghĩa là:
Dãy của N số phức: $x_0, ..., x_{N-1}$ được biến đổi thành chuỗi của N số phức $X_0, ..., X_{N-1}$ bởi công thức sau đây:

$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn}$ với $k = 0, ..., N-1$.
với e là cơ số lôgarit tự nhiên, i là đơn vị ảo ($i^2 = -1$), và π là pi. Phép biến đổi đôi khi được ký hiệu bởi $\mathbf{F}$, như sau $\mathbf{X} = \mathbf{F\{x\}}$ hoặc $\mathbf{F(x)}$ hoặc $\mathbf{Fx}$.

Khi tính trực tiếp từ định nghĩa trên đòi hỏi $O(N^2)$ phép tính: có N số X_k cần tính, để tính mỗi số cần tính một tổng N số hạng. Một FFT là một phương pháp để tính cùng kết quả đó trong $O(NlogN)$ phép tính.

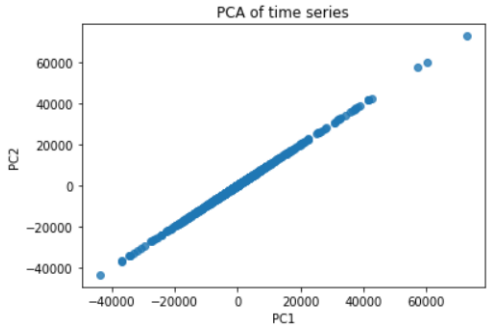
- Sau khi biến đổi Fourier xong, ta tiến hành ép các dữ liệu từ biến chứa các dữ liệu đã được tách ra thành 3 cột lại thành 1 dữ liệu thống nhất bằng phương pháp flatten.

2 Giai đoạn xử lý:

- Giai đoạn xử lý gồm 2 bước là: giảm số chiều và phân cụm.

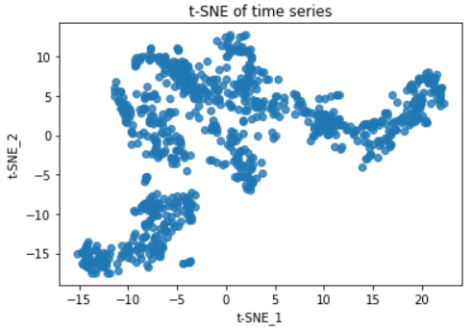
2.1 Giảm số chiều.

- Sau các bước xử lý dữ liệu thô, ta tiến hành hạ số chiều của các file xuống còn 2 chiều, bằng PCA của thư viện scikit-learn. Sau đó, tiến hành trực quan hóa dữ liệu có được để quan sát.



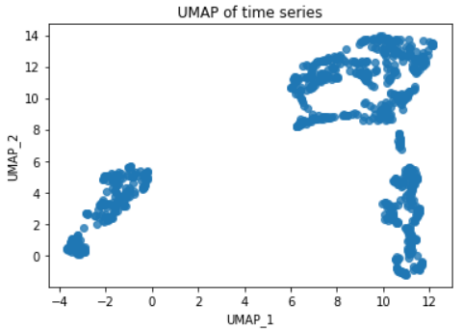
Hình 3: Hình vẽ mô phỏng các Time-Series khi giảm số chiều xuống còn 2 bằng thuật toán PCA.

- Bằng cách khác, chúng ta có thể tiến hành hạ số chiều của các file xuống còn 2 chiều bằng t-SNE của thư viện scikit-learn. Sau đó, vẽ hình để quan sát.



Hình 4: Hình vẽ mô phỏng các Time-Series khi giảm số chiều xuống còn 2 bằng thuật toán t-SNE.

- Ngoài ra, chúng ta có thể tiến hành hạ số chiều của các file xuống còn 2 chiều bằng UMAP của thư viện scikit-learn. Sau đó, vẽ hình để quan sát.



Hình 5: Hình vẽ mô phỏng các Time-Series khi giảm số chiều xuống còn 2 bằng thuật toán UMAP.

- Qua đó có thể thấy sự khác biệt rõ ràng của cả 3 thuật toán giảm số chiều:

+ **Đối với thuật toán PCA**, bằng cách xác định siêu phẳng (the hyperplane) nằm gần dữ liệu nhất và sau đó chiếu dữ liệu lên siêu phẳng đó trong khi vẫn giữ lại hầu hết các biến thể trong tập dữ liệu. Trục giải thích lượng phương sai tối đa trong tập huấn luyện được gọi là Các thành phần chính.

+ **Đối với thuật toán t-SNE**, lấy tập dữ liệu có chiều cao và giảm nó thành đồ thị có chiều thấp để giữ lại nhiều thông tin ban đầu. Nó làm như vậy bằng cách cung cấp cho mỗi điểm dữ liệu một vị trí trong bản đồ hai hoặc ba chiều. Kỹ thuật này tìm các cụm trong dữ liệu do đó đảm bảo rằng một phép nhúng duy trì ý nghĩa trong dữ liệu. t-SNE làm giảm kích thước trong khi cố gắng giữ cho các trường hợp tương tự gần nhau và các trường hợp khác nhau cách xa nhau.

+ **Đối với thuật toán UMAP**, là một phương pháp giảm kích thước phi tuyến tính, nó rất hiệu quả để hiển thị các cụm hoặc nhóm điểm dữ liệu và các điểm gần nhau tương đối của chúng.

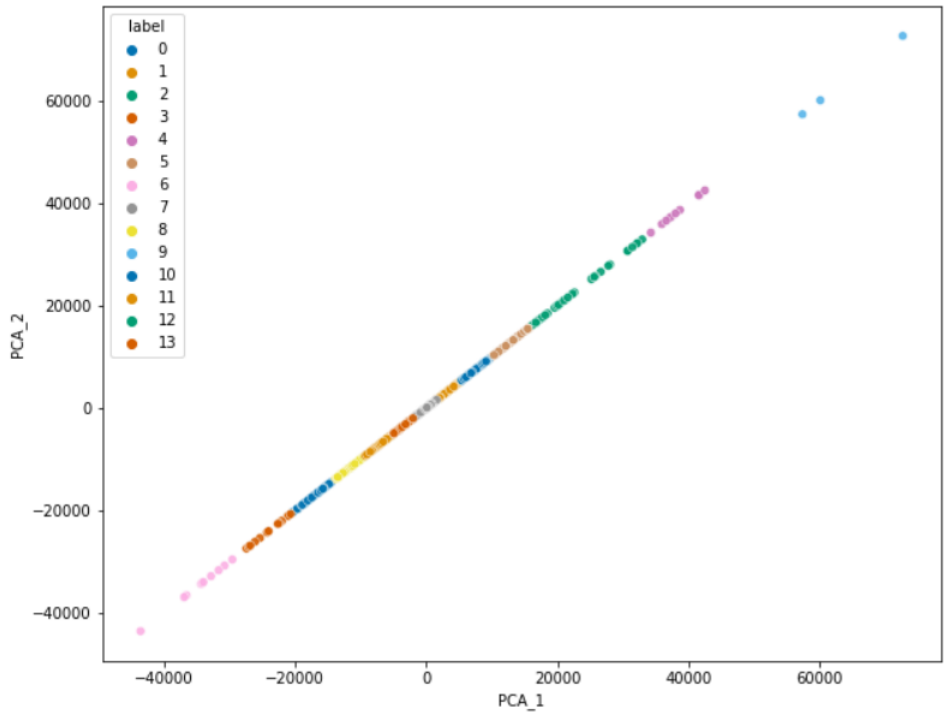
Sự khác biệt đáng kể với t-SNE là khả năng mở rộng, nó có thể được áp dụng trực tiếp cho các ma trận thưa thớt, do đó loại bỏ đi được sự cần thiết phải áp dụng bất kỳ phương pháp giảm Thứ nguyên nào như PCA hoặc Truncated SVD (Phân hủy giá trị đơn lẻ) như một bước tiền xử lý trước.

Nói một cách đơn giản, UMAP tương tự như t-SNE nhưng với tốc độ xử lý có lẽ cao hơn, do đó, nhanh hơn và có thể hiển thị tốt hơn.

2.2 Phân cụm.

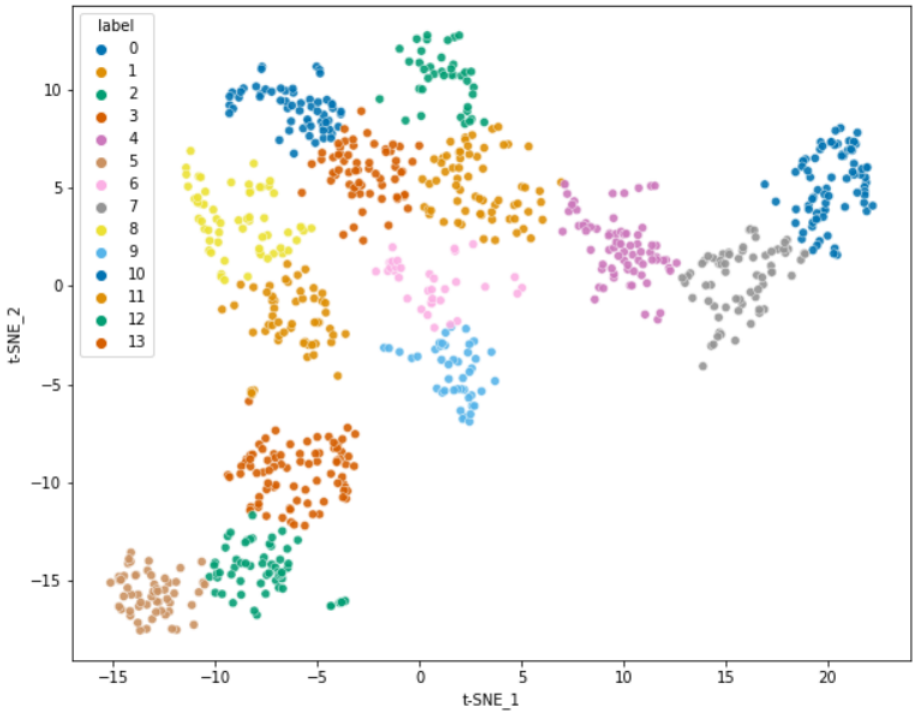
- Bước tiếp theo, sau khi đã giảm số chiều của file Time-Series, ta tiến hành K-Means để phân loại các Time-Series theo 14 cụm. Từ đó ta tiến hành trực quan hóa dữ liệu có được.

- **K-Means thông qua việc hạ số chiều bằng PCA**,



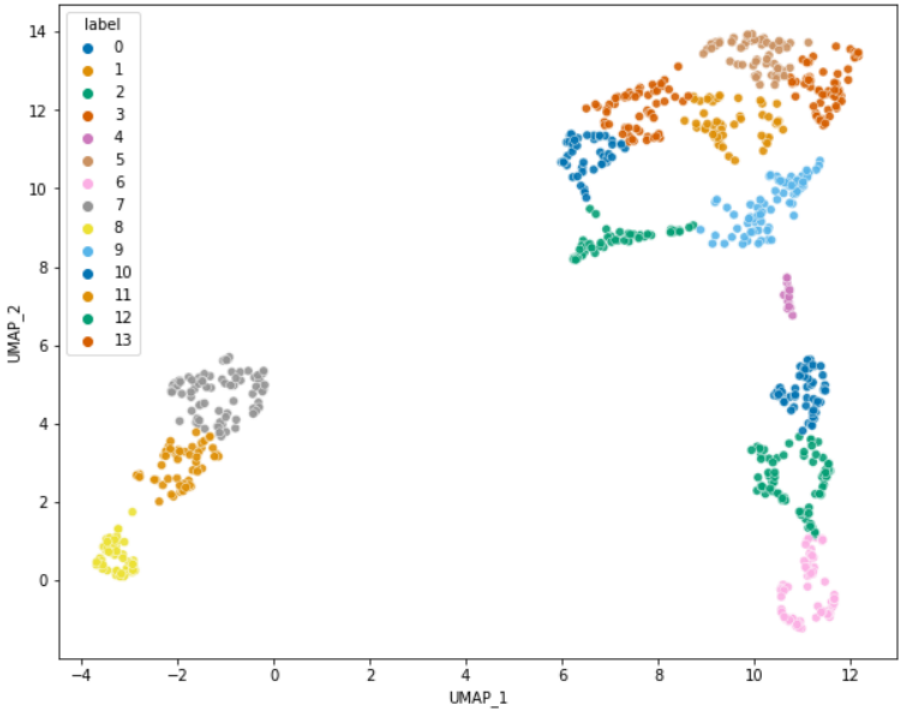
Hình 6: Hình vẽ mô phỏng các Time-Series khi thực hiện K-Means qua việc giảm số chiều xuống còn 2 bằng thuật toán PCA.

- **K-Means thông qua việc hạ số chiều bằng t-SNE**,



Hình 7: Hình vẽ mô phỏng các Time-Series khi thực hiện K-Means qua việc giảm số chiều xuống còn 2 bằng thuật toán t-SNE.

- **K-Means thông qua việc hạ số chiều bằng UMAP**,



Hình 8: Hình vẽ mô phỏng các Time-Series khi thực hiện K-Means qua việc giảm số chiều xuống còn 2 bằng thuật toán UMAP.

- Thuật toán K-Means là thuật toán thuộc loại học không giám sát(tức dữ liệu không được gán nhãn sẵn).

- Thuật toán phân cụm K-Means là một phương pháp được sử dụng trong phân tích tính chất cụm của dữ liệu. Nó đặc biệt được sử dụng nhiều trong khai phá dữ liệu và thống kê. Nó phân vùng dữ liệu thành k cụm khác nhau. Giải thuật này giúp chúng ta xác định được dữ liệu của chúng ta nó thực sử thuộc về nhóm nào.

- Ý tưởng của thuật toán phân cụm K-Means:

1. Khởi tạo K điểm dữ liệu trong bộ dữ liệu và tạm thời coi nó là tâm của các cụm dữ liệu của chúng ta.
2. Với mỗi điểm dữ liệu trong bộ dữ liệu, tâm cụm của nó sẽ được xác định là 1 trong K tâm cụm gần nó nhất.
3. Sau khi tất cả các điểm dữ liệu đã có tâm, tính toán lại vị trí của tâm cụm để đảm bảo tâm của cụm nằm ở chính giữa cụm.
4. Lặp đi lặp lại bước 2 và bước 3 cho tới khi vị trí của tâm cụm không thay đổi hoặc tâm của tất cả các điểm dữ liệu không thay đổi.

3 Công việc phân công:

- Trịnh Ngọc Hiến:
+ Xử lý phần giai đoạn tiền xử lý.
+ Viết báo cáo.

- Nguyễn Ngô Trung Hậu:
+ Xử lý phần giai đoạn xử lý.
+ Thuyết trình.