Đại học quốc gia thành phố Hồ Chí Minh
Trường Đại Học Khoa học Tự Nhiên
Khoa Công nghệ thông tin

Ứng dụng mô hình Hidden Markov cho bài toán nhận diện giọng nói

Vũ Đăng Hoàng Long (18120203)
Nguyễn Huỳnh Đại Lợi (18120198)
Huỳnh Long Nam (18120212)
Phạm Nhật Minh (18120209)
Nguyễn Bảo Long (18120201)

Đồ án 01
Môn: Thống kê máy tính và ứng dụng

Giáo viên
Thầy Vũ Quốc Hoàng

Thành phố Hồ Chí Minh, 2021

Đánh giá đồ án

Tên thành viên	Công việc	Đánh giá
Vũ Đăng Hoàng Long	Quản lý nhóm, phân việc và đảm bảo chất lượng theo yêu cầu, hỗ trợ các thành viên.	Hoàn thành
Huỳnh Long Nam	Tìm hiểu để làm demo, và thử nghiệm	Hoàn thành
Nguyễn Huỳnh Đại Lợi	Tìm hiểu để làm demo, và thử nghiệm	Hoàn thành
Phạm Nhật Minh	Tìm hiểu lý thuyết về HMM và xử lý âm thanh	Hoàn thành
Nguyễn Bảo Long	Tìm hiểu để làm demo, và thử nghiệm	Hoàn thành

Mục lục

Báo cáo

1. GIỚI THIỆU BÀI TOÁN

Đối với con người, âm thanh chính là phương thức giao tiếp hiệu quả và thông dụng nhất. Cũng chính vì thế mà nhận diện giọng nói là một chủ đề hữu ích, có nhiều ứng dụng. Dễ dàng thấy được tính ứng dụng này khi mà hiện nay các trợ lý ảo trở nên rất phổ biến, hay các ứng dụng nhập liệu bằng giọng nói,…

Bài toán nhận diện giọng nói nói chung là bài toán tìm cách giúp cho máy tính có thể hiểu được con người và giọng nói, từ đó có các phản hồi hợp lý. Trong đồ án này, với mục đích chính là tìm hiểu về cách ứng dụng mô hình Hidden Markov, bài toán chính nhóm chọn thực hiện là nhận diện chữ số từ âm thanh cô lập (isolated digit recognition). Đây là bài toán theo nhóm đánh giá là không quá phức tạp, là nền tảng cho các toán lớn hơn như nhận diện chữ từ giọng nói hay chuyển từ giọng đọc thành văn bản.

Bài toán: nhận diện chữ số từ âm thanh (isolated digit recognition).

Dữ liệu đầu vào: một đoạn âm thanh ngắn chứa giọng đọc một số bất kỳ từ 0 tới 9.

Dữ liệu ra: cho biết số từ âm thanh này.

2. VẤN ĐỀ VỚI BÀI TOÁN

Trong bài toán này chúng ta có các vấn đề sau cần giải quyết:

- *Dữ liệu âm thanh đầu vào là một chuỗi số rất dài*: với một file âm thanh 1s với sample rate là 8000hz thì dữ liệu là chuỗi 8000 số. Việc học mô hình trên chuỗi này là không hiệu quả, do vậy chúng ta cần phải áp dụng các kỹ thuật xử lý âm thanh để giảm kích thước dữ liệu đầu vào lại cũng như lấy được các đặc trưng quan trọng giúp mô hình dễ học được hơn.
- *Dữ liệu có thể bị nhiễu nặng:* đây là một vấn đề nan giải hầu hết mọi file thu âm đều bị. Để đơn giản hóa bài toán, nhóm sẽ không làm việc với file bị lẫn tạp âm quá nặng.
- *Nguồn dữ liệu*: với nguồn dữ liệu nhóm chọn 2 nguồn:
    - FSDD (Jakobovski/free-spoken-digit-dataset: A free audio dataset of spoken digits. Think MNIST for audio. (github.com)): đây là dữ liệu tiếng anh, gồm 3000 mẫu giọng đọc từ 6 người, mỗi người sẽ đọc mỗi số từ 0-9 50 lần (50x10x6 = 3000).
    - Speech Commands (https://www.wolfram.com/language/12/machine-learning-for-audio/classify-spoken-digits.html?product=mathematica): đây là dữ liệu của Google bao gồm giọng đọc các số và một số lệnh đơn (left, right,…). Phiên bản nhóm sử dụng do Wolfram cung cấp, tách ra dữ liệu giọng đọc số ra từ dữ liệu gốc. Dữ liệu này có 10000 mẫu cho tập huấn luyện và 1000 mẫu cho tập test, được đọc bởi 997 người.

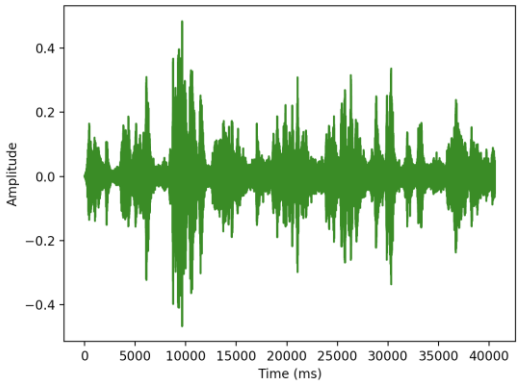
— *Kiến trúc mô hình:* dữ liệu âm thanh ở đây không đơn giản, và mô hình nhóm sử dụng cũng không thể đơn giản được. Nếu chọn một mô hình HMM có kiến trúc bất kỳ, nhóm sẽ không thể đảm bảo được chắc chắn mô hình sẽ học được như ý muốn. Kiến trúc mô hình ở đây nhóm sẽ tham khảo từ bài báo [3].

— *Không dễ để tìm ví dụ tham khảo trên mạng*: do hiện nay bài toán này có thể giải quyết khá hiệu quả và đơn giản với mạng neural (và thậm chí từ những năm 2000 đã bắt đầu sử dụng mạng neural cho bài toán này), do vậy khá ít người cài đặt và upload mô hình này lên mạng mà không dùng mạng neural. Do vậy hầu hết nhóm phải tham khảo mô hình từ các bài báo và tự cài lại, và không phải kiến trúc nào cũng cài đặt được. Nhóm sẽ cố gắng thử nghiệm nhiều nhất có thể để đúc kết một số kết luận và đánh giá các hướng phát triển mô hình trong tương lai.

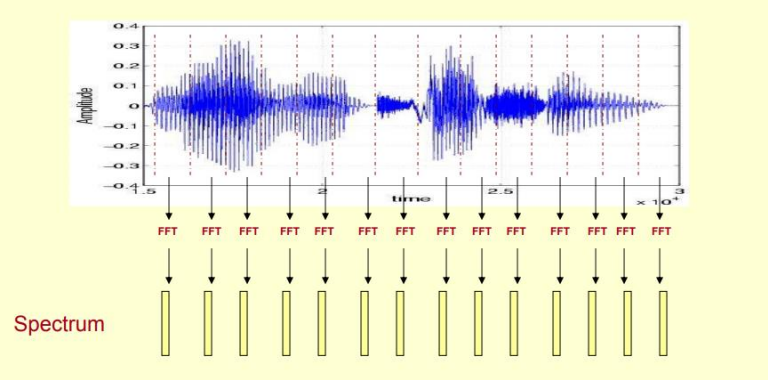
3. Xử lý âm thanh – Lý thuyết Mel Frequency Cepstral Coefficients (MFCC)

MFCC là một cách để trích xuất các đặc trưng giọng nói thường được sử dụng trong các model nhận dạng giọng nói (Automatic Speech Recognition) hay phân loại giọng nói (Speech Classification). MFCC sẽ cho ra kết quả là các hệ số (coefficients) của cepstral từ Mel filter trên phổ lấy được từ các file âm thanh chứa giọng nói.

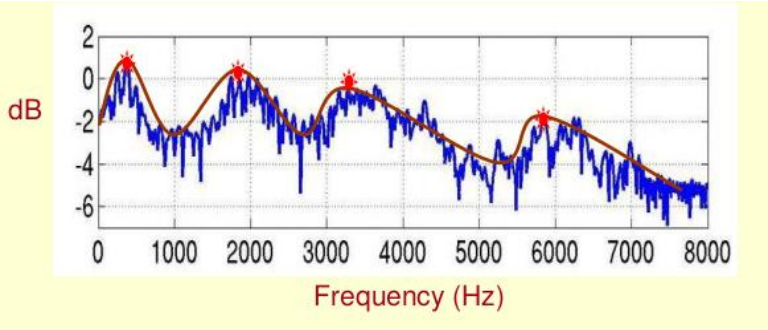
Dữ liệu giọng nói thì thường sẽ được biểu diễn dưới dạng hai chiều (x, y) với x là thời gian (time) theo milliseconds (ms) và y là biên độ (amplitude). Trong đó những giá trị trên y được gọi là speech signal.



Đầu tiên, speech signal được biến đổi thành âm phổ (spectrum) dùng thuật toán Fast Fourier Transform.

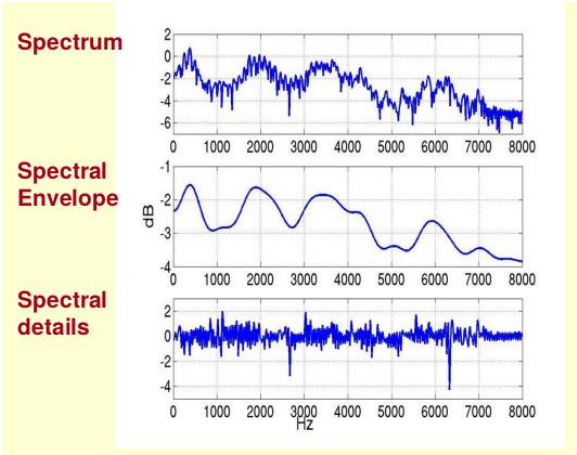


Kết quả của việc biến đổi này, tức là spectrum, được biểu diễn dưới dạng hai chiều (x', y') với x' là tần số (Hz) và y' là cường độ (dB).



Ở hình trên, các điểm màu đỏ được gọi là Formants, là nơi có các tần số áp đảo (dominant), mang đặc tính của âm thanh. Đường màu đỏ gọi là Spectral Envelopes. Mục tiêu chính của ta là lấy được đường màu đỏ này.

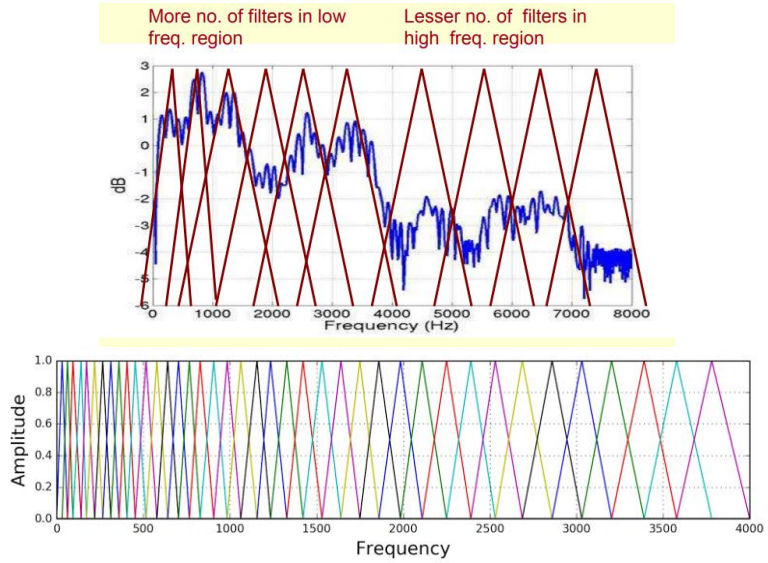
Gọi spectrum là X[k] có hai thành phần là spectral envelopes H[k] và spectral details E[k]



Để tách được H[k], ta cần phải lấy logarithm của spectrum và lấy phần ở tần số thấp (low frequency):

$$X[k] = H[k] * E[k]$$
$$\Leftrightarrow log(X[k]) = log(H[k]) + log(E[k])$$

Các thí nghiệm trên tai người cho thấy rằng tai người hoạt động như một bộ lọc, chỉ tập trung vào một phần thay vì hết cả spectral envelopes. Từ đó, bộ lọc Mel-Frequency Filter lấy ý tưởng này ra đời.



Qua thực nghiệm nghiên cứu tiếng nói, các phương trình chuyển đổi giữa mel (m) và hertz (f) được đề ra như sau:

$$m = 2595 log_{10}(1 + \frac{f}{700})$$
$$f = 700(10^{m/2595} - 1)$$

Mel-frequency filter:

$$\begin{aligned} H_m(k) &= 0 && \text{iff } k < f(m-1) \\ &= \frac{k - f(m-1)}{f(m) - f(m-1)} && \text{iff } f(m-1) \le k < f(m) \\ &= 1 && \text{iff } k = f(m) \\ &= \frac{f(m+1) - k}{f(m+1) - f(m)} && \text{iff } f(m) < k \le f(m+1) \\ &= 0 && \text{iff } k > f(m+1) \end{aligned}$$

Sau khi áp dụng MFCC, ta sẽ sử dụng Inverse Fast Fourier Transform lên logarithm của spectrum:

$$IFFT(log(X[k])) = IFFT(log(H[k]) + log(E[k]))$$
$$\Leftrightarrow x[k] = h[k] + e[k]$$

Với x[k] là nghịch đảo của spectrum, do IFFT là nghịch đảo của FFT, từ đó dẫn đến tên gọi cepstrum.

Cepstrum được biểu diễn dưới dạng hai chiều (x", y"), nhưng với giá trị và các tên khác: y" là magnitude (không có đơn vị) và x" là quefrency (đại lượng đối nghịch của frequency) (ms).

MFCC chính là các hệ số cepstral từ cepstrum, với giá trị thông thường là 12 hệ số của y''.

4. Lý thuyết mô hình Hidden Markov

4.1 Discrete-Time Markov Model

Xét một hệ thống luôn ở trong 1 trong N trạng thái khác nhau $S_1, S_2, S_3, \dots, S_N$. Hệ thống thay đổi trạng thái sau mỗi chu kì nhất định theo các xác suất của trạng thái. Gọi thời gian mà trạng thái thay đổi là $t = 1,2, \dots$ và trạng thái tại thời điểm t là q_t.

Để mô hình hóa hệ thống trên một cách toàn diện, ta cần thông tin về trạng thái hiện tại và các trạng thái trước. Xét trường hợp xích Markov thời gian rời rạc, ta chỉ cần biết trạng thái trước và trạng thái hiện tại:

$$P(q_t = j | q_{t-1} = i, q_{t-2} = k, \dots) = P(q_t = j | q_{t-1} = i)$$

Xét những hệ thống mà vế phải của phương trình trên độc lập với thời gian, ta có các thành phần sau:

— Xác suất chuyển trạng thái a_{ij}:

$$a_{ij} = P(q_t = j | q_{t-1} = i), 1 \le i, j \le N$$

Với

$$a_{ij} \ge 0, \qquad \forall i, j$$
$$\sum_{j=1}^{N} a_{ij} = 1, \qquad \forall i$$

— Ma trận chuyển:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{bmatrix}_{N \times N}$$

— Phân phối ban đầu:

$$\boldsymbol{\pi} = \begin{bmatrix} \pi_1 = P(q_1 = 1) \\ \pi_2 = P(q_1 = 2) \\ \vdots \\ \pi_N = P(q_1 = N) \end{bmatrix}_{N \times 1}$$

Với:

$$\sum_{i=1}^{N} \pi_i = 1$$
$$\pi_i = P(q_1 = i), \qquad 1 \le i \le N$$

Quá trình trên được gọi là một mô hình Markov với kết quả là các trạng thái tại một thời điểm nhất định, với mỗi trạng thái tương ứng với một sự kiện nhất định. Mô hình được đặc trưng bởi ma trận chuyển A và trạng thái ban đầu π

4.2 DISCRETE-TIME HIDDEN MARKOV MODEL

Mô hình Markov trên khó áp dụng vào các bài toán thực tế do có nhiều giới hạn. Vì vậy, ta sẽ mở rộng mô hình ra thành mô hình Markov ẩn, với sự khác biệt là thay vì mỗi trạng thái có một kết quả xác định thì giờ kết quả của trạng thái sẽ tuân theo mô hình xác suất. Mỗi trạng thái giờ sẽ sinh ra một quan sát o_t dựa theo một hàm xác suất nào đó. Hàm sinh quan sát là:

$$B = \{b_j(o_t)\}_{j=1}^{N}$$

với hàm xác suất của mỗi trạng thái j là $b_j(o_t) = P(o_t|q_t = j)$.

4.3 KIỂU KIẾN TRÚC HMM

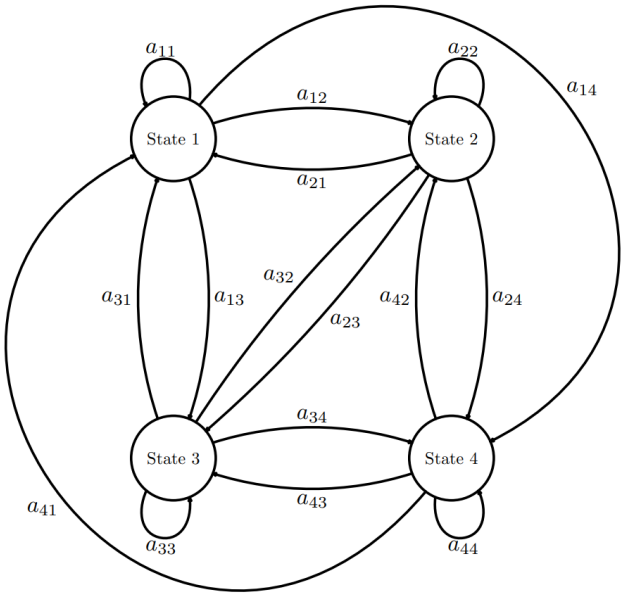
Có nhiều kiến trúc HMM khác nhau, được xác định bởi ma trận chuyển A.

4.3.1 ERGODIC HMM

Kiến trúc thường gặp nhất là ergodic (fully connected). Trong kiến trúc này, mọi trạng thái đều có thể chuyển qua tất cả các trạng thái khác. Với một mô hình ergodic có N=4, ta có:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}_{4\times 4}$$

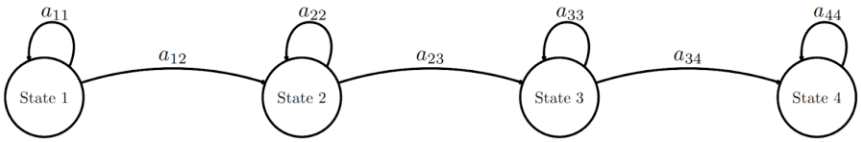
Với $0 < a_{ij} < 1$



Hình 1: Hình ảnh minh họa cho kiến trúc ergodic với số trạng thái ẩn N=4.

4.3.2 LEFT-RIGHT HMM

Mô hình trái-phải (mô hình Bakis) là một dạng mô hình mà các quan sát được thực hiện lần lượt có thứ tự từ trái sang phải– giống với đặc tính của tiếng nói. Do đó, mô hình này phù hợp cho các ứng dụng speech recognition.



Hình 2: Hình ảnh minh họa cho kiến trúc left-right với 4 trạng thái ẩn và khoảng cách chuyển là 1.

Tính chất của mô hình:

— Không thể chuyển về trạng thái trước.

$$a_{ij} = 0, \qquad j < i$$

— Khoảng cách chuyển trạng thái thường được giới hạn bởi một độ dài nhất định, thường là 2 hoặc 3.

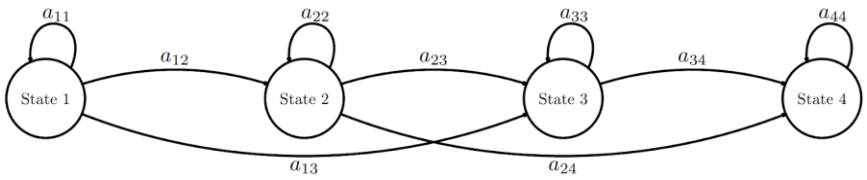
$$a_{ij} = 0, \qquad j > i + \Delta$$

— Có trạng thái cuối.

$$a_{NN} = 1, \qquad a_{Nj} = 0, \ \ j < N$$

Với mô hình ở hình 2, $\Delta = 1$, ma trận chuyển là

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}_{4\times 4}$$



Hình 3: Hình ảnh minh họa cho kiến trúc left-right với 4 trạng thái ẩn và khoảng cách chuyển là 2.

Với hình 3, $\Delta = 2$, ma trận chuyển có dạng

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}_{4\times 4}$$

4.4 Các thành phần của mô hình HMM

1. Số trạng thái ẩn N. Trạng thái ở thời điểm t gọi là q_t.
2. Phân phối ban đầu.

$$\boldsymbol{\pi} = \{\pi_i\}_{i=1}^{N}, \qquad \pi_i = P(q_1 = i)$$

3. Ma trận chuyển.

$$\mathbf{A} = [a_{ij}]$$
$$a_{ij} = P(q_{t+1} = j | q_t = i), \qquad 1 \le i, j \le N$$

4. Hàm sinh quan sát

$$\mathbf{B} = \{b_j(\mathbf{o}_t)\}_{j=1}^{N}$$

với hàm xác suất cho mỗi trạng thái j

$$b_j(\mathbf{o}_t) = P(\mathbf{o}_t | q_t = j)$$

Cuối cùng, ta ký hiệu một mô hình HMM hoàn chỉnh gồm $\boldsymbol{\pi}, \mathbf{A}, \mathbf{B}$:

$$\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$$

4.5 Các bài toán lớn

1. Bài toán so khớp (likelihood): Với chuỗi quan sát $\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_T)$ và mô hình $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ tính xác suất chuỗi quan sát xảy ra với mô hình? Nói cách khác, vấn đề này là bài toán tính $P(\mathbf{O}|\lambda)$.

2. Bài toán giải mã (decoding): Với chuỗi quan sát $\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_T)$ và mô hình $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$, làm sao chọn được chuỗi trạng thái $\mathbf{q} = (q_1, q_2, \dots, q_T)$ tối ưu nhất (phù hợp nhất với chuỗi quan sát)?
3. Bài toán huấn luyện mô hình (learning): Cho chuỗi quan sát $\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_T)$, làm sao chọn các tham số của $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ để tối đa hóa $P(\mathbf{O}|\lambda)$?

Bài toán đầu tiên được xem là bài toán nhận diện: với nhiều mô hình khác nhau, mô hình nào ứng với một chuỗi quan sát nhất định nhất (với ứng dụng hiện tại: nhận diện từ đang được nói).

Bài toán thứ hai tìm các trạng thái ẩn của mô hình. Tuy nhiên, không có câu trả lời "đúng" nhất cho bài toán này, nên thường câu trả lời sẽ được lựa chọn theo các tiêu chí "tối ưu" nhất.

Bài toán thứ ba là bài toán huấn luyện: với một chuỗi quan sát nhất định (ứng với một từ), tạo một mô hình cho chuỗi quan sát này. Đây là bài toán quan trọng nhất, ảnh hưởng lớn đến việc áp dụng HMM vào các ứng dụng thực tế. Qua quá trình huấn luyện, các tham số của mô hình sẽ dần được thích ứng với dữ liệu huấn luyện – đồng nghĩa với việc mô hình đáp ứng tốt với hiện tượng thực tế.

Trong bài toán nhận diện giọng nói của nhóm, để giải quyết sẽ cần giải quyết 2 trong 3 bài toán trên: bài toán huấn luyện để học các mô hình và bài toán likelihood để áp dụng mô hình, nhận diện giọng nói.

4.5.1 BÀI TOÁN LIKELIHOOD

Yêu cầu của bài toán này là tìm xác suất của chuỗi quan sát $\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_T)$ của mô hình $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$. Vì các quan sát này độc lập với nhau và với t, nên xác suất chuỗi quan sát này xảy ra với chuỗi trạng thái $\mathbf{q}$ bất kì được tính bằng tích:

$$P(\mathbf{O}|\mathbf{q}, \mathbf{B}) = b_{q_1}(\mathbf{o}_1) \cdot b_{q_2}(\mathbf{o}_2) \dots b_{q_T}(\mathbf{o}_T)$$

với xác suất của chuỗi trạng thái $\mathbf{q}$ là

$$P(\mathbf{q}|\mathbf{A}, \boldsymbol{\pi}) = \pi_{q_1} \cdot a_{q_1 q_2} \cdot a_{q_2 q_3} \cdot \ldots \cdot a_{q_{T-1} q_T}$$

Xác suất $\mathbf{O}$ và $\mathbf{q}$ cùng xảy ra là tích của 2 xác suất trên

$$\begin{aligned} P(\mathbf{O}, \mathbf{q}|\lambda) &= P(\mathbf{O}|\mathbf{q}, B) \cdot P(\mathbf{q}|\mathbf{A}, \boldsymbol{\pi}) \\ &= \pi_{q_1} b_{q_1}(\mathbf{o}_1) a_{q_1 q_2} b_{q_2}(\mathbf{o}_2) \cdot \ldots \cdot a_{q_{T-1} q_T} b_{q_T}(\mathbf{o}_T) \\ &= \pi_{q_1} \prod_{t=2}^{T} a_{q_{t-1} q_t} b_{q_t}(\mathbf{o}_t) \end{aligned}$$

Như yêu cầu đề bài, để tính $P(\mathbf{O}|\lambda)$, ta tính tổng xác suất $\mathbf{O}$ và $\mathbf{q}$ cùng xảy ra trên tất cả các trạng thái $\mathbf{q}$. Ta có:

$$
\begin{aligned}
P\left(\mathbf{O}|\lambda\right) &= \sum_{\text{all } \mathbf{q}} P\left(\mathbf{O}|\mathbf{q}, B\right) \cdot P(\mathbf{q}|\mathbf{A}, \boldsymbol{\pi}) \\
&= \sum_{q_1, q_2, \ldots, q_T} \pi_{q_1} b_{q_1}(\mathbf{o}_1) a_{q_1 q_2} b_{q_2}(\mathbf{o}_2) \cdot \ldots \cdot a_{q_{T-1} q_T} b_{q_T}(\mathbf{o}_T) \\
&= \sum_{q_1, q_2, \ldots, q_T} \pi_{q_1} \prod_{t=2}^{T} a_{q_{t-1} q_t} b_{q_t}(\mathbf{o}_t)
\end{aligned}
$$

Quá trình tính toán của công thức trên được mô phỏng như sau:

- Tại thời điểm $t = 1$, khởi tạo chương trình với trạng thái q_1 với xác suất π_{q_1} và hình thành quan sát $\mathbf{o}_1$ với xác suất $b_{q_1}(\mathbf{o}_1)$
- Tại thời điểm $t + 1$, trạng thái chuyển từ q_1 sang q_2 với xác suất $a_{q_1 q_2}$ và hình thành quan sát $\mathbf{o}_2$ với xác suất $b_{q_2}(\mathbf{o}_2)$.

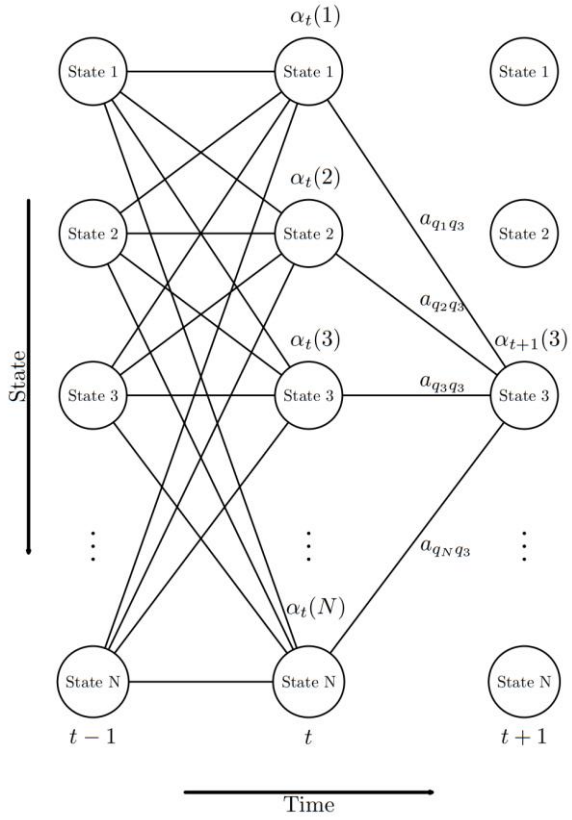
Quá trình tiếp tục đến thời điểm cuối cùng T, trạng thái chuyển từ q_{T-1} sang q_T với xác suất $a_{q_{T-1} q_T}$ và hình thành quan sát $\mathbf{o}_T$ với xác suất $b_{q_T}(\mathbf{o}_T)$.

Tuy nhiên, quá trình tính toán như trên không khả thi do khối lượng phép tính quá lớn: cần $(2T - 1)N^T$ phép nhân và N^T phép cộng. Ngay cả với khối lượng trạng thái và quan sát nhỏ, lấy ví dụ $N = 5$ và $T = 10$, ta cần thực hiện $(2 \cdot 10 - 1) \cdot 5^{10} = 1.8 \cdot 10^8$ phép nhân và $5^{10} - 1 = 9.7 \cdot 10^6$ phép cộng. Để giải quyết vấn đề này, ta có thể áp dụng Forward Algorithm:

Forward algorithm
Xét 1 biến forward $\alpha_t(i)$

$$\alpha_t(i) = P\left(\mathbf{o}_1 \mathbf{o}_2 \ldots \mathbf{o}_t, q_t = i|\lambda\right)$$

với trạng thái i và thời gian t. Ta có $\alpha_t(i)$ chính là xác suất của chuỗi quan sát $\mathbf{o}_1 \mathbf{o}_2 \ldots \mathbf{o}_t$ khi ở trạng thái i và thời gian t. Dùng quy nạp để tính $\alpha_t(i)$ theo hình 4.



Hình 4: Sơ đồ tính toán của thuật toán Forward.

Theo hình, $\alpha_{t+1}(i)$ được tính bằng tổng tất cả các tích giữa biến forward của N trạng thái ở thời điểm t , xác suất chuyển trạng thái a_{ij} tương ứng và xác suất của trạng thái $b_j(\mathbf{o}_{t+1})$. Thuật toán gồm 4 bước:

1. Khai báo:

$$\begin{aligned} \text{Set } t &= 1; \\ \alpha_1(i) &= \pi_i b_i(\mathbf{o}_1), \quad 1 \le i \le N \end{aligned}$$

2. Quy nạp

$$\alpha_{t+1}(j) = b_j(\mathbf{o}_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij}, \quad 1 \le j \le N$$

3. Cập nhật thời gian: nếu $t < T$, quay lại bước (2). Ngược lại, qua bước (4).
4. Kết thúc với

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Với thuật toán này, ta chỉ cần thực hiện $N(N+1)(T-1)+N$ phép nhân và $N(N-1)(T-1)$ phép cộng. Xét với ví dụ $N=5$ và $T=10$, ta chỉ cần thực hiện $5(5+1)(10-1)+5 = 275$ phép nhân và $5(5-1)(10-1) = 180$ phép cộng. Số lượng phép tính giảm rất nhiều so với phương pháp tính truyền thống.

4.5.2 BÀI TOÁN HUẤN LUYỆN

Bài toán này giải quyết vấn đề ước tính các tham số $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ của mô hình, nghĩa là tính

$$\lambda^* = arg\max_{\lambda}[P(\mathbf{O}|\lambda)]$$

Với một quan sát **O**, tìm lambda* sao cho P(O|lambda) lớn nhất. Trong ba bài toán, bài toán này là bài toán khó xử lý nhất vì chưa có cách tối ưu nào để tính được các tham số tốt nhất cho mô hình.

Một phương pháp thường được dùng để xử lý bài toán này là thuật toán Baum-Welch (còn gọi là expectation maximization). Thuật toán Baum-Welch có những ưu điểm như sau, so với những phương pháp khác để xử lý bài toán này:

- Thuật toán ổn định về mặt số học - xác suất P(O|lambda) không giảm với mỗi lần lặp
- Baum-Welch hội tụ về một cực trị lân cận (local optima)
- Baum-Welch hội tụ tuyến tính.

Re-estimation algorithm

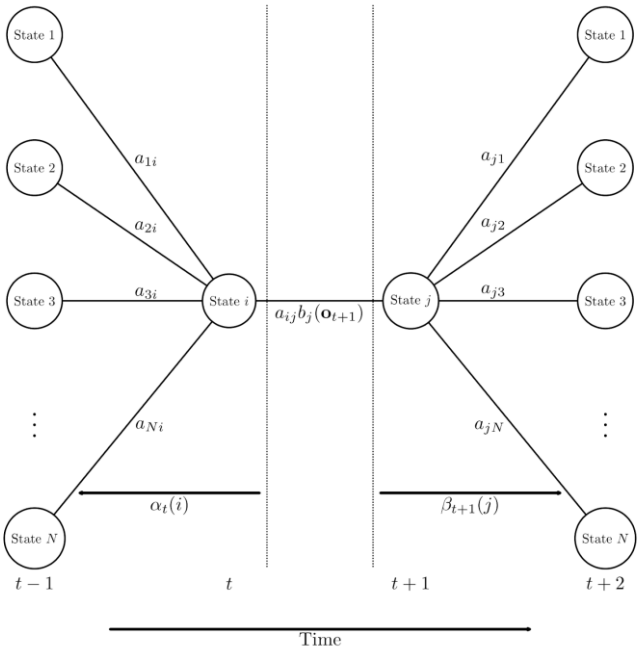
Mô hình lambda có 3 tham số: phân phối chuyển trạng thái A, phân phối xác suất ban đầu pi và phân phối xác suất quan sát B.

Gọi xi_t(i,j) là xác suất ở trạng thái i ở thời điểm t và ở trạng thái j trong thời điểm t+1 với mô hình lambda và chuỗi quan sát O.

Ta có

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j|\mathbf{O}, \lambda) = \frac{P(q_t = i, q_{t+1} = j, \mathbf{O}|\lambda)}{P(\mathbf{O}|\lambda)}$$

Mô hình thỏa điều kiện công thức trên:



Hình 5: Sơ đồ tính toán cho thuật toán Baum-Welch.

Gọi $\beta_t(i)$ là biến backward, đại diện cho xác suất chuỗi quan sát 1 phần từ $t+1$ đến hết T với trạng thái i ở thời gian t và mô hình λ.

$$\beta_t(i) = P\left(\mathbf{o}_{t+1}\mathbf{o}_{t+2}\ldots\mathbf{o}_T|q_t = i, \lambda\right)$$

Ta có:

$$\begin{aligned}\xi_t(i,j) &= \frac{\alpha_t(i)a_{ij}b_j(\mathbf{o}_{t+1})\beta_{t+1}(j)}{P\left(\mathbf{O}|\lambda\right)} \\ &= \frac{\alpha_t(i)a_{ij}b_j(\mathbf{o}_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(\mathbf{o}_{t+1})\beta_{t+1}(j)}\end{aligned}$$

Với $\gamma_t(i)$ đại diện cho xác suất ở trạng thái i vào thời điểm t với chuỗi quan sát O và mô hình lambda như bài toán 2, ta thu được:

$$\gamma_t(i) = P\left(q_t = i|\mathbf{O}, \lambda\right) = \sum_{j=1}^{N} P\left(q_t = i, q_{t+1} = j|\mathbf{O}, \lambda\right) = \sum_{j=1}^{N}\xi_t(i,j)$$

Tính tổng $\gamma_t(i)$, ta thu được 1 đại lượng đại diện cho số lần (dự kiến) chuyển đến trạng thái i, cũng chính là số lần chuyển trạng thái từ i (nếu không tính trạng thái cuối cùng T).

Tương tự với $\xi_t(i)$, ta thu được số lần (dự kiến) chuyển từ trạng thái i sang trạng thái j.

Ta có:

$\gamma_1(i) =$ xác suất bắt đầu ở trạng thái i

$\sum_{t=1}^{T-1} \gamma_t(i)$ = số lần (dự kiến) chuyển trạng thái từ trạng thái i trong O

$\sum_{t=1}^{T-1} \xi_t(i,j)$ = số lần (dự kiến) chuyển trạng thái từ i sang j trong O

Từ đó, ta tính được công thức cho pi và A

π_i = tần số dự kiến ở trạng thái i tại thời điểm t = 1

$$= \gamma_1(i)$$

$$= \frac{\alpha_1(i)\beta_1(i)}{\sum_{i=1}^{N} \alpha_1(i)\beta_1(i)}$$

$$= \frac{\alpha_1(i)\beta_1(i)}{\sum_{i=1}^{N} \alpha_T(i)}$$

$$A_{ij} = \frac{\text{số lần (dự kiến)chuyển trạng thái từ i sang j}}{\text{số lần (dự kiến) chuyển trạng thái từ trạng thái i}}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$= \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i)\beta_t(i)}$$

Với

$$P(\mathbf{O}|,\lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) = \sum_{i=1}^{N} \alpha_T(i)$$

$$P(\mathbf{O}, q_t = i|,\lambda) = \alpha_t(i)\beta_t(i)$$

$$P(\mathbf{O}, q_t = i, q_{t+1} = j|,\lambda) = \alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)$$

Tuy nhiên, với bài toán nhận diện giọng nói, dữ liệu đầu vào là một số liên tục chứ không phải số rời rạc, do vậy B cần là phân phối đại diện cho các quan sát liên tục - ở đây phổ biến nhất

là phân phối gauss. Kế đến B có khả năng phát sinh quan sát từ nhiều phân phối khác nhau, do vậy ta cần kết hợp cả phân phối trộn. Tổng quát lại, ta có hàm xác suất cho mỗi trạng thái j là:

$$b_j(\mathbf{o}_t) = \sum_{k=1}^{M} c_{jk} b_{jk}(\mathbf{o}_t), \quad j = 1, 2, \ldots, N$$

Với b_{jk} là hàm gauss đa biến với vector mean μ_{jk} và ma trận hiệp phương sai $\sum_{jk}$

$$b_{jk}(\mathbf{o}_t) = \mathcal{N}\left(\mathbf{o}_t, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}\right)$$

Nếu chỉ có 1 trạng thái j và một hàm phân phối (k=1):

$$\begin{aligned} c_j &= 1 \\ \boldsymbol{\mu}_j &= \frac{1}{T}\sum_{t=1}^{T} \mathbf{o}_t \\ \boldsymbol{\Sigma}_j &= \frac{1}{T}\sum_{t=1}^{T} \left(\mathbf{o}_t - \boldsymbol{\mu}_j\right)\left(\mathbf{o}_t - \boldsymbol{\mu}_j\right)' \end{aligned}$$

(Kí hiệu chuyển vị ở đây là ‘ để không nhầm với biến T)

Tuy nhiên, trong thực tế có nhiều trạng thái khác nhau, có nhiều hàm phân phối khác nhau (k!=1) và ta không thể gán một chuỗi quan sát cho một trạng thái khi chưa biết được chuỗi trạng thái ẩn. Do xác suất của mỗi chuỗi quan sát bằng tổng xác suất của tất cả các chuỗi trạng thái khả thi với chuỗi quan sát đó, mỗi vector quan sát ot đều ảnh hưởng đến giá trị xác suất của trạng thái j. Thay vì gán mỗi vector quan sát cho một trạng thái nhất định, mỗi vector quan sát sẽ được gán cho tất cả các trạng thái và có trọng số là xác suất của mô hình ở trạng thái đó, tính theo các hàm phân phối khi ghi nhận được quan sát đó. Xác suất này cho trạng thái j và số lượng hàm k (có tổng cộng M hàm) được tính như sau:

$$\gamma_t(j,k) = \frac{\alpha_t(j)\beta_t(j)}{\sum_{i=1}^{N} \alpha_t(j)\beta_t(j)} \cdot \frac{c_{jk}\mathcal{N}\left(\mathbf{o}_t, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}\right)}{\sum_{k=1}^{M} c_{jk}\mathcal{N}\left(\mathbf{o}_t, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}\right)}$$

C_jk là tỉ lệ số lần (dự kiến) mô hình đạt trạng thái j với hàm thứ k / số lần (dự kiến) mô hình đạt trạng thái j

$$c_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T}\sum_{k=1}^{M} \gamma_t(j,k)}$$

Muy_jk và sigma_jk được tính bằng cách thêm trọng số bằng xác suất ở trạng thái j với hàm thứ k khi quan sát ot vào công thức ở phần trên:

$$\boldsymbol{\mu}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)\mathbf{o}_t}{\sum_{t=1}^{T} \gamma_t(j,k)}$$

$$\boldsymbol{\Sigma}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)\left(\mathbf{o}_t - \boldsymbol{\mu}_j\right)\left(\mathbf{o}_t - \boldsymbol{\mu}_j\right)'}{\sum_{t=1}^{T} \gamma_t(j,k)}$$

Tuy nhiên, các công thức trên chỉ hoạt động hiệu quả với một mẫu. Để phù hợp hơn với các ứng dụng thực tế và tránh overfitting, ta sẽ thay đổi công thức như sau:

Gọi chuỗi R quan sát là $\mathrm{O} = \left[\mathrm{O}^{(1)}, \mathrm{O}^{(2)} \ldots, \mathrm{O}^{(R)}\right]$ với $\mathbf{O}^{(r)} = \left(\mathbf{o}_1^{(r)}, \mathbf{o}_2^{(r)}, \ldots, \mathbf{o}_{T_r}^{(r)}\right)$ là chuỗi quan sát thứ r và T_r là độ dài của chuỗi đó. Ta sẽ có xác suất mới để tối ưu hóa là:

$$P\left(\mathbf{O}|\lambda\right) = \prod_{r=1}^{R} P\left(\mathbf{O}^{(r)}|\lambda\right)$$

Với mỗi mẫu (chuỗi quan sát), các biến $\alpha_t(j)$, $\beta_t(j)$, $\gamma_t(j,k)$ và $\xi_t(j,k)$ được tính cho mỗi mẫu, sau đó quy trình ước tính được thực hiện trên tất cả các giá trị thu được, lặp lại đến khi nào tìm được cực trị hoặc đạt đến một giới hạn số lần nhất định.

Với mỗi mẫu, ta tính tham số ~P* (tt Viterbi), sau đó tính tổng các tham số này lại để ra được xác suất toàn cục. Khi xác suất này không tăng, ta tìm được cực trị.

5. Cài đặt trên Python

Để cài đặt mô hình này, nhóm sử dụng thư viện Pomegranate. Mặc dù tìm kiếm trên google rất nhiều người giới thiệu và khuyên dùng thư viện hmmlearn, tuy nhiên lúc chạy thử nhóm thấy thư viện này có vấn đề là nhóm không thể chọn kiến trúc của mô hình HMM được, trong khi đây là điều nhóm cho rằng quan trọng nhất. Do đó nhóm sử dụng thư viện này.

Notebook nhóm cài đặt để phục vụ cho demo được lưu ở [đây]. Lưu ý rằng notebook này chỉ để phục vụ mục đích demo nên có thể có một số phần cài đặt không nằm trong đây, để có thể coi kỹ hơn cài đặt, vui lòng đọc đến các phần sau để lấy link đến các notebook thí nghiệm của nhóm.

5.1 Sử dụng thư viện Pomegranate để tạo và sử dụng mô hình HMM

Pomegranate là thư viện giúp tạo và sử dụng các mô hình xác suất với cú pháp đơn giản như scikit-learn cũng các API dễ sử dụng giúp xây dựng được mô hình phức tạp mà không

phải quan tâm quá nhiều tới vấn đề cài đặt. Bên cạnh đó thư viện này được cài đặt trên nền Cython nên cho được tốc độ chạy nhanh.

Để biết cách sử dụng và làm việc với mô hình HMM của pomegrante, tham khảo notebook chính thức khá chi tiết của tác giả (link).

Về tổng quan thì cách xây dựng khá đơn giản như chơi xếp hình: tạo các object State cho mỗi trạng thái và thêm vào, sau đó thêm các connection mong muốn hoặc thêm hẳn ma trận chuyển (nhóm khuyến khích cách đầu vì thường mô hình làm việc với HMM khá phức tạp). Sau đó gọi bake() để thư viện tự kiểm tra và điều chỉnh tham số, xác suất để đảm bảo không lỗi. Cuối cùng ta gọi fit(X) để mô hình học.

5.2 CÁC THÀNH PHẦN CỦA CHƯƠNG TRÌNH

Trong phần này nhóm sẽ nói một số hàm chính có thể sử dụng chung, những hàm còn lại phục vụ cho mục đích cụ thể như tổ chức dữ liệu, test kết quả với một tập dữ liệu,… nhóm sẽ không đề cập.

Đọc một file vào và thực hiện tiền xử lý: `full_mfcc_from_file(<path>, trim=True)`.

(ở một số notebook cũ do chưa sửa, nhóm sử dụng hàm `extract_mfcc(<path>)` để đọc, hàm này thì không thực hiện tiền xử lý mà chỉ đọc và rút trích thành đặc trưng MFCC)

Hàm `left_right_GMMHMM(x_dim, n_states=10, n_modals=9, random=0)` sẽ tự generate một mô hình HMM với kiến trúc left-right như hình 5, với phân phối của quan sát là gauss trộn với 9 mixtures. Tuy nhiên do bug với thư viện mà nhóm chưa thể train được mô hình với phân phối trộn. Do đó nhóm hầu như chỉ thử nghiệm với `n_modals=1` là chính.

Hàm `train_GMMHMM(dataset, input_dim, n_hidden_state, n_gauss_modal)` sẽ tự tạo mô hình như trên, nhưng ở đây có kèm dataset để tự học và return mô hình đã huấn luyện.

Ngoài ra nhóm còn nhiều hàm bổ trợ khác, thầy có thể tham khảo notebook để rõ thêm (mục 9).

6. CÁC MÔ HÌNH SẼ THỬ NGHIỆM VÀ CÁC CÁCH ĐÁNH GIÁ

Ở đây nhóm sẽ thử 3 kiến trúc chính với các thay đổi tham số như sau:

- Số trạng thái ẩn: 10 states, 5 states và 3 states
- Số mixtures: 1 (tức thuần gauss), 3 và 9. Tuy nhiên do thư viện không thể chạy được với mixture nên nhóm chỉ thử 3 với 9 trong thử nghiệm đầu, các thử nghiệm về sau nhóm sẽ chỉ thử với 1 gauss.

Về cơ bản thì theo paper nhóm đọc thì kiến trúc tốt nhất là 10 hidden states và 9 mixtures. Tuy nhiên nhóm vẫn thử nghiệm thêm các số khác vì mong muốn đánh giá, xem xét khả năng học của model nếu cho số state ít.

Bên cạnh đó, do nhóm sử dụng 2 dataset nhóm cũng sẽ có các thiết lập cách test để đánh giá khả năng học của mô hình. Có 4 bài test tất cả cho mỗi mô hình:

- Train trên 40% data FSDD, sau đó test lần lượt trên 60% của data FSDD và test 100% data Wolfram. Mục đích của bài test này là để đánh giá khả năng generalize trên bộ data mà mô hình được học và khả năng generalize với data thực tế, mở rộng hơn (do data FSDD chỉ có 6 speaker nên nhiều khả năng mô hình này mô hình luôn giọng nói của speaker, dẫn đến việc fail khi gặp người khác).
- Train trên 100% data FSDD, sau đó test 100% data Wolfram. Mục đích của bài test này là để đánh giá khả năng năng generalize với data thực tế nếu được cho nhiều data hơn, mở rộng hơn (nguyên do vẫn là do FSDD chỉ có 6 speaker).
- Train trên 30% data FSDD và 30% data Wolfram, sau đó test lần lượt trên phần còn lại của data. Lần này nhóm cho mô hình tiếp cận với data Wolfram với mong muốn data sẽ đa dạng hơn và mô hình tránh được overfit tốt hơn.
- Train trên 60% data FSDD và 60% data Wolfram, sau đó test lần lượt trên phần còn lại của data. Đây là bài test cuối cùng với số lượng lên đến gần 8000 mẫu, nhóm hy vọng với lượng data này là đủ để đánh giá toàn diện cho mô hình.

7. Kết quả thử nghiệm đầu và đánh giá

Đây là thử nghiệm huấn luyện đơn giản, dữ liệu ở đây không tiền xử lý, chỉ rút trích thẳng đặc trưng MFCC bằng thư viện librosa.

7.1 Thử nghiệm 1: train trên 40% data FSDD, test 100% Wolfram

Kiến trúc mô hình	Thời gian train	Test FSDD (60%)	Test Wolfram (100%)
10 state, 9 mixture	2min37s	54.06%	10.76%
5 state, 9 mixture	1min22s	38.83%	10.02%
5 state, 3 mixture	39.7s	38.94%	10.02%

7.2 Thử nghiệm 2: train trên 100% data FSDD, test 100% Wolfram

Kiến trúc mô hình	Thời gian train	Train FSDD (100%)	Test Wolfram (100%)
10 state, 9 mixture	11min44s	75.10%	15.96%
5 state, 9 mixture	3min11s	56.17%	15.02%
5 state, 3 mixture	2min1s	56.10%	15.35%

7.3 Thử nghiệm 3: train trên 30% data FSDD + 30% Wolfram, test phần còn lại

Kiến trúc mô hình	Thời gian train	Train Mix (30%+30%)	Test Mix (70%+70%)
10 state, 9 mixture	11min5s	44.32%	35.40%
5 state, 9 mixture	6min3s	31.25%	27.64%
5 state, 3 mixture	2min15s	30.62%	27.21%

7.4 Thử nghiệm 4: train trên 60% data FSDD + 60% Wolfram, test phần còn lại

Kiến trúc mô hình	Thời gian train	Train Mix (60%+60%)	Test Mix (40%+40%)
10 state, 9 mixture	20min10s	40.15%	35.31%
5 state, 9 mixture	14min14s	30.68%	28.84%
5 state, 3 mixture	6min31s	30.93%	29.15%

7.5 Nhận xét và đánh giá

Đây là cài đặt của nhóm sau khi đọc paper và cài theo kiến trúc, tuy nhiên kết quả chạy tệ hơn nhóm tưởng. Tuy nhiên sau khi nhóm bật verbose trong quá trình train, nhóm phát hiện rằng mô hình bị lỗi trong lúc học nhưng tác giả đã supress, không cho báo hay văng lỗi ra nên làm nhóm tưởng rằng mô hình này không có khả năng học.

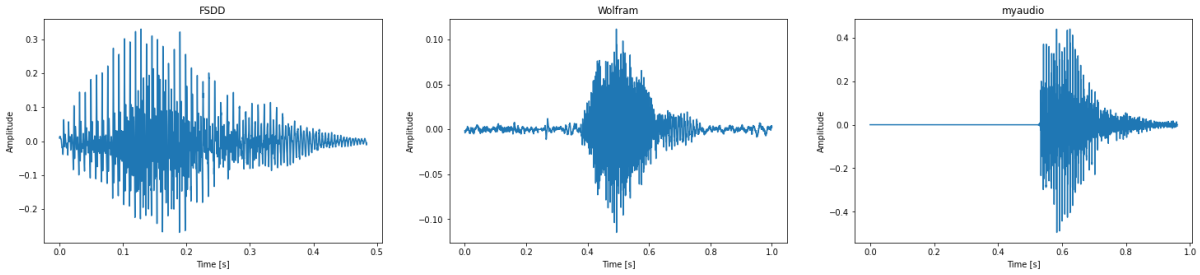
Ở một thử nghiệm khác thì nhóm tách đặc trưng MFCC bằng thư viện python_speech_features, kết quả ở test 1 mô hình 10-9 (gọi tắt cho 10 hidden states và 9 mixtures modal) thì mô hình fit rất tốt, cho ra kết quả 99% trên tập FSDD, tuy nhiên chỉ được 15% trên tập Wolfram.

Cả 2 thử nghiệm trên do đã làm rất lâu nên nhóm không có lưu lại. Nhìn chung thì vấn đề rất có thể nằm ở data FSDD khi mô hình có thể đạt kết quả tốt trên data này nhưng

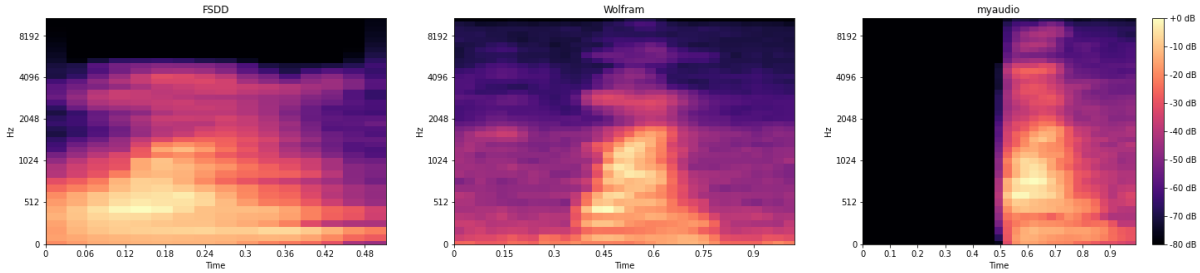
8. Khám phá dữ liệu âm thanh và tiền xử lý

Trong phần này nhóm sẽ thử khám phá dữ liệu âm thanh để điều tra và cải thiện mô hình. Do nhóm không có kiến thức nhiều về xử lý tín hiệu số và âm thanh nên với nhóm bước này sẽ khá quan trọng để khám phá điều gì ảnh hưởng tới mô hình. Và cũng với lý do đó, các kỹ thuật tiền xử lý ở đây chỉ là trực quan, cảm nhận của nhóm chứ không phải kỹ thuật đã được chứng minh là có hiệu quả.

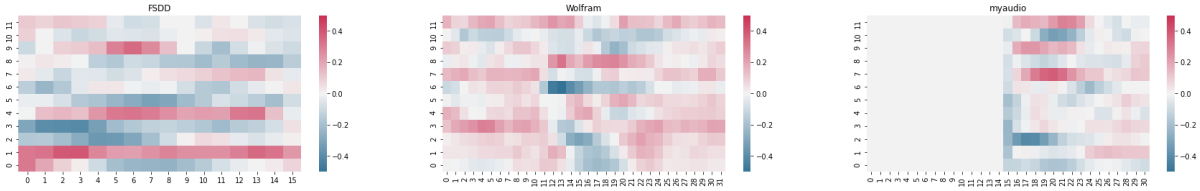
Đầu tiên chúng ta sẽ lần lượt xem xét waveform, spectrogram và MFCC của sampel audio đọc số 1 (one) từ FSDD, Wolfram và file giọng đọc của sinh viên:



Hình 6: Waveform



Hình 7: Spectrogram



Hình 8: MFCC

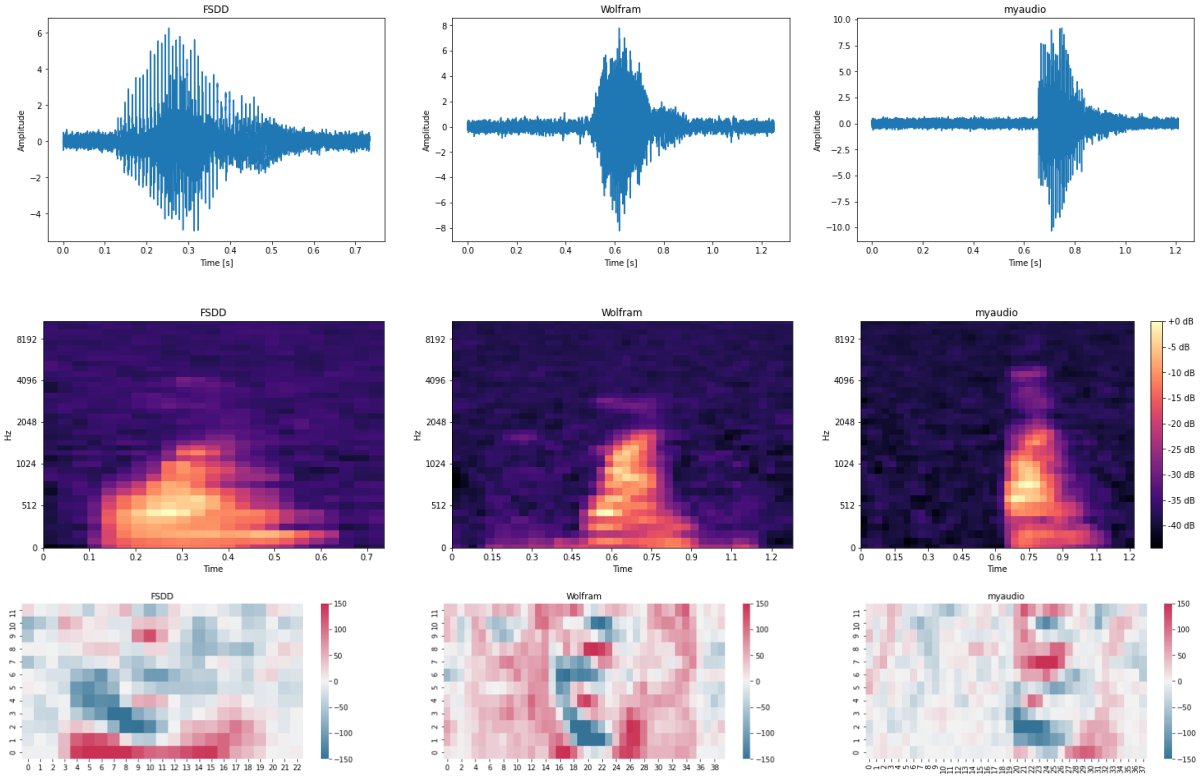
Nhận xét:

- Nhìn vào waveform, sự tương đồng không quá rõ rệt, tuy nhiên nhìn vào spectrogram ta dễ dàng thấy được điểm tương đồng (chú ý kỹ vùng sáng).
- Data FSDD được cắt sát, không có khoảng trống, nhưng 2 data kia thì không như vậy.
- Data Wolfram có noise.

Với quan sát trên, dễ dàng thấy được vì sao mà mô hình của chúng ta bị overfit hoặc không thể fit được. Tới đây nhóm sẽ thực hiện tiền xủ lý như sau:

- Thêm khoảng trắng ở 2 đầu của mọi file audio, nhóm mong muốn rằng model của mình cũng phải ghi nhận rằng audio của mình có trạng thái trống không có audio.
- Thêm noise vào toàn audio. Do mô hình nàu học unsupervise, nếu không có noise, mô hình sẽ cho rằng mọi file audio của mình có điểm chung là khoảng trắng, từ đó mà overfit, không học được.
- Scale lại waveform trước khi rút trích đặc trưng MFCC. Điều này chưa có chứng minh ý nghĩa, nhưng về lý luận thì nhóm cho rằng audio các file chênh lệch khá nhiều (có file âm lượng to, file âm lượng thấp,…). Khi tiền xử lý lại thì nhóm thấy rằng ảnh hưởng của noise trong file Wolfram nó giảm rõ rệt.

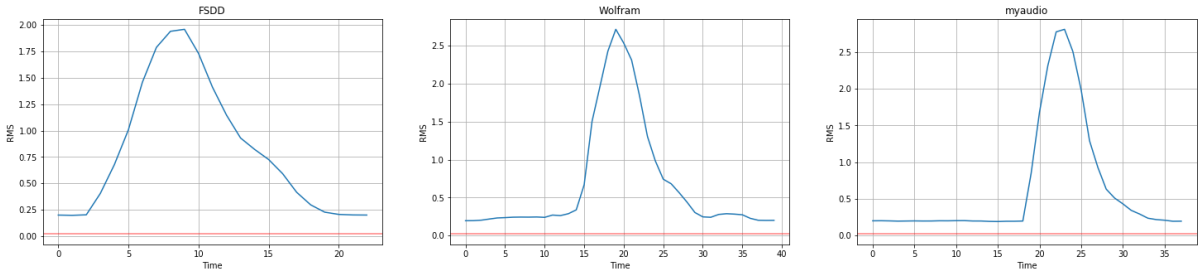
Sau khi tiền xử thi các biểu đồ trông như sau.



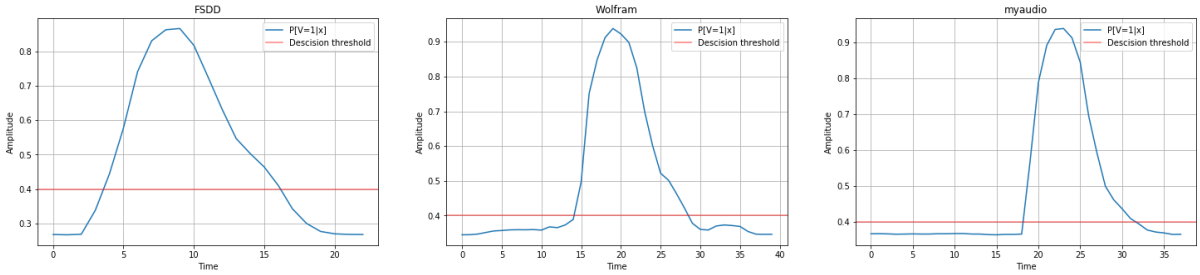
Dễ dàng thấy được đặc trưng được làm nổi bật rõ rệt trong spectrogram! Bên cạnh đó, theo nhiều người cho rằng, việc chỉ dùng 12 đặc trưng thuần của MFCC là chưa đủ, nếu chúng ta thêm đạo hàm của các đặc trưng này (tức thêm 12 đặc trưng nữa thành 24 đặc trưng) thì kết quả được cải thiện rất rõ rệt. Đây cũng là tiền xử lý được sử dụng trong thử nghiệm exp002 tới exp009.

Tuy nhiên, cho rằng mô hình không thể capture được khoảng trắng trong âm thanh dễ dàng, nhóm tiến hành bước xử lý nâng cao khác là cắt đi khoảng trắng trong âm thanh!

Lấy RMS của spectrogram ta thấy được điểm nhô này



Lấy softmax của các điểm nhô, ta thấy rằng lấy ở khoảng giá trị 0.4 là ta cắt được khoảng trắng.



Đây cũng là tiền xử lý nâng cao cho thử nghiệm exp10 tới exp15

9. KẾT QUẢ CÁC THỬ NGHIỆM VÀ ĐÁNH GIÁ SAU CÙNG

Đầu tiên, nhóm thí nghiệm khả năng generalize trên cùng tập với tập train và khả năng generalize với tập không được học (test số 1).

Nhận xét:

- Kết quả đánh giá tập train và tập test FSDD giảm đi (từ 99% xuống còn 87%) nhưng bù lại khả năng generalize trên tập không được học tăng lên đáng kể (15% lên 48%)
- Đạo hàm bậc 1 của MFCC là 1 đặc trưng quan trọng (exp006 có độ chính xác giảm đi một nửa)
- Việc sử dụng nhiều hơn 12 hệ số MFCC không giúp cải thiện
- Việc sử dụng phân phối gauss trộn thay cho gauss thuần túy không giúp cải thiện đáng kể kết quả trên tập test wolfram
- Việc cắt bỏ khoảng không âm không giúp mô hình 10 hidden cải thiện, tuy nhiên lại giúp mô hình chỉ với 5 hay 3 hidden cải thiện đáng kể, bắt kịp với mô hình 10 hidden

	Đánh giá tập train (40% FSDD)	**Đánh giá tập test 60% FSDD**	**Đánh giá tập test 100% Wolfram**
24 đặc trưng MFCC (12 thuần + **12 đạo hàm**) Tiền xử lý: chuẩn hóa + thêm khoảng trắng + thêm noise Phân phối của quan sát: Gauss đường chéo			
Exp002-10 hidden	87.50%	85.11%	47.98%
Exp002-5 hidden	56.00%	56.28%	28.93%
Exp002-3 hidden	42.08%	39.72%	18.63%
12 đặc trưng MFCC (12 thuần) Tiền xử lý: chuẩn hóa + thêm khoảng trắng + thêm noise Phân phối của quan sát: Gauss đường chéo			
Exp006-10 hidden	62.17%	52.83%	22.49%
Exp006-5 hidden	42.75%	42.94%	17.13%
Exp006-3 hidden	31.42%	32.61%	17.24%
24 đặc trưng MFCC (12 thuần + **12 đạo hàm**) Tiền xử lý: chuẩn hóa + thêm khoảng trắng + thêm noise **+ cắt khoảng không âm** Phân phối của quan sát: Gauss đường chéo			
Exp010-10 hidden	85.92%	83.78%	53.74%

Exp010-5 hidden	85.50%	84.78%	54.33%
Exp010-3 hidden	79.08%	78.06%	46.08%
40 đặc trưng MFCC (20 thuần **+ 20 đạo hàm**) Tiền xử lý: chuẩn hóa + thêm khoảng trắng + thêm noise **+ cắt khoảng không âm** Phân phối của quan sát: Gauss đường chéo			
Exp012-10 hidden	85.00%	84.67%	53.71%
Exp012-5 hidden	81.25%	78.17%	47.05%
Exp012-3 hidden	75.08%	76.83%	44.17%
24 đặc trưng MFCC (12 thuần **+ 12 đạo hàm**) Tiền xử lý: chuẩn hóa + thêm khoảng trắng + thêm noise **+ cắt khoảng không âm** Phân phối của quan sát: Gauss đường chéo**, trộn 3 đỉnh**			
Exp018-10 hidden	88.00%	86.39%	54.47%
Exp018-5 hidden	86.58%	84.78%	52.19%
Exp018-3 hidden	75.08%	76.00%	47.37%
24 đặc trưng MFCC (12 thuần **+ 12 đạo hàm**) Tiền xử lý: chuẩn hóa + thêm khoảng trắng + thêm noise **+ cắt khoảng không âm** Phân phối của quan sát: Gauss đường chéo**, trộn 9 đỉnh**			
Exp020-10 hidden	90.83%	86.56%	54.89%
Exp020-5 hidden	87.33%	84.39%	50.93%
Exp020-3 hidden	83.42%	79.72%	41.93%

Kế đến, nhóm thí nghiệm khả năng generalize khi cho phép học trên cả 2 tập (tất nhiên học với rất ít dữ liệu chỉ 30%, test số 3)

Nhận xét:

- Kết quả đánh giá tập tập test FSDD giảm đi, tập test wolfram lại tăng lên, tiệm cận với nhau
- Cải thiện được độ chính xác lên đến 75% với phân phối quan sát là gauss trộn 3 đỉnh

	Đánh giá tập train (30% + 30%)	**Đánh giá tập test 70% FSDD**	**Đánh giá tập test 70% Wolfram**
24 đặc trưng MFCC (12 thuần **+ 12 đạo hàm**) Tiền xử lý: chuẩn hóa + thêm khoảng trắng + thêm noise Phân phối của quan sát: Gauss đường chéo			
Exp004-10 hidden	66.68%	71.52%	65.00%
Exp004-5 hidden	48.98%	49.76%	47.82%
Exp004-3 hidden	28.99%	30.52%	29.05%
12 đặc trưng MFCC (12 thuần) Tiền xử lý: chuẩn hóa + thêm khoảng trắng + thêm noise Phân phối của quan sát: Gauss đường chéo			
Exp008-10 hidden	31.82%	29.71%	30.79%
Exp008-5 hidden	26.35%	25.86%	25.32%
Exp008-3 hidden	21.53%	18.81%	21.36%
24 đặc trưng MFCC (12 thuần **+ 12 đạo hàm**)			

Tiền xử lý: chuẩn hóa + thêm khoảng trắng + thêm noise **+ cắt khoảng không âm** Phân phối của quan sát: Gauss đường chéo			
Exp011-10 hidden	74.85%	71.05%	73.85%
Exp011-5 hidden	69.58%	64.95%	69.76%
Exp011-3 hidden	59.96%	55.90%	59.51%
40 đặc trưng MFCC (20 thuần **+ 20 đạo hàm**) Tiền xử lý: chuẩn hóa + thêm khoảng trắng + thêm noise **+ cắt khoảng không âm** Phân phối của quan sát: Gauss đường chéo			
Exp013-10 hidden	73.66%	67.48%	73.28%
Exp013-5 hidden	69.52%	67.24%	69.23%
Exp013-3 hidden	61.97%	61.81%	62.00%
24 đặc trưng MFCC (12 thuần **+ 12 đạo hàm**) Tiền xử lý: chuẩn hóa + thêm khoảng trắng + thêm noise **+ cắt khoảng không âm** Phân phối của quan sát: Gauss đường chéo**, trộn 3 đỉnh**			
Exp016-10 hidden	75.87%	73.05%	75.20%
Exp016-5 hidden	69.71%	67.67%	68.69%
Exp016-3 hidden	56.47%	60.05%	55.94%
24 đặc trưng MFCC (12 thuần **+ 12 đạo hàm**) Tiền xử lý: chuẩn hóa + thêm khoảng trắng + thêm noise **+ cắt khoảng không âm** Phân phối của quan sát: Gauss đường chéo**, trộn 9 đỉnh**			
Exp021-10 hidden	77.30%	75.67%	75.46%
Exp021-5 hidden	70.83%	69.14%	69.35%
Exp021-3 hidden	61.25%	56.43%	60.75%

Như vậy, qua các đánh giá trên ta có thể thấy bước khám phá và tiền xử lý đã có thể giúp mô hình cải thiện được rất nhiều trong kết quả (từ 15% lên đến 75%).

10. LINK CÁC NOTEBOOK THÍ NGHIỆM

Notebook dùng để demo:

https://colab.research.google.com/drive/1sGneIuFpHd5MVHQYM8-t1GJfknT5yIGu

https://colab.research.google.com/drive/1mHiC6Fpn6xq-yA6P3ZcaHZqfmXP7815g
(phiên bản rút gọn, không có code train data giúp demo nhanh hơn)

Experiment 002: thí nghiệm 1 mixtures trên 24 đặc trưng MFCC, test số 1

https://colab.research.google.com/drive/13EiiMN2kc_5gShydgktE8zVlEdiN5wIT#scrollTo=lbl8PZfk3qrX

Experiment 003: tiếp tục exp002 với test số 2

https://colab.research.google.com/drive/1lSHnh3Cs70VPyS0grykk29H9efMQIax0#scrollTo=LXnRu87vi4II

Experiment 004: tiếp tục exp002 với test số 3

https://colab.research.google.com/drive/1AUABrE5sRlOSP07ZdNSsNCHub5jGYEWM#scrollTo=JdS1CTOeA44e

Experiment 005: tiếp tục exp002 với test số 4

https://colab.research.google.com/drive/1yi2Xv2B81pD__TLxgx2xGldf2AEHPzIW#scrollTo=xsOJoIodiHt1

Experiment 006: thí nghiệm 1 mixtures trên 12 đặc trưng MFCC, test số 1

https://colab.research.google.com/drive/1rg3NGeN3hL7w433z134eHdzg7qaHC05c#scrollTo=xsOJoIodiHt1

Experiment 007: tiếp tục exp006 với test số 2

https://colab.research.google.com/drive/1Ti9lRET04uMIEJI-dxOx5RSffUjmbZ3m#scrollTo=ikasEdh-I_wD

Experiment 008: tiếp tục exp006 với test số 3

https://colab.research.google.com/drive/1lSH5czkmM85Gf2WRKEAgkgSl3E301-sr#scrollTo=m2RHG_z9nEsf

Experiment 009: tiếp tục exp006 với test số 4

https://colab.research.google.com/drive/1YkFP06goHnbozLj-eEeUT7Yk0v2e5ZhV#scrollTo=ikasEdh-I_wD

Experiment 010: thí nghiệm có tiền xử lý và khám phá file âm thanh, 24 đặc trưng MFCC, test số 1

https://colab.research.google.com/drive/1uxrOCCwYmvBKR_lIBh0lrmg_5izw2Tso#scrollTo=xsOJoIodiHt1

Experiment 011: tiếp tục exp010 với test số 3

https://colab.research.google.com/drive/1eXnzbiwT8e06xLCmuFzREjwhUZmQi0Hb?usp=sharing

Experiment 012: thí nghiệm có tiền xử lý và khám phá file âm thanh, 40 đặc trưng MFCC, test số 1

https://colab.research.google.com/drive/1Pp2E4b9DTSsED9726pTCZPtkcFZgLEW2?usp=sharing

Experiment 013: tiếp tục exp012 với test số 3

https://colab.research.google.com/drive/1i_b4sfvCGYB4vw1iqYuWGLKAe6fgAZq5?usp=sharing

Experiment 014: tiếp tục exp010 với test số 4 (có thay đổi tỉ lệ về 50%)

https://colab.research.google.com/drive/1neQwKzJTIdD-XaztkZ9dnVZrAGGou6tO?usp=sharing

Experiment 015: tiếp tục exp012 với test số 4 (có thể đổi tỉ lệ về 50%)

https://colab.research.google.com/drive/1Yl_8fl1ilp9quH8VOgYpnuDMHH6xkPeX?usp=sharing

Experiment 016: thí nghiệm 3 mixtures, test số 3

https://colab.research.google.com/drive/1EuTAjKhSZF01pnzSyytvyFPR-e6R9kD8

Experiment 017: tiếp tục exp016 với test số 4

https://colab.research.google.com/drive/1brIKGILcNIXadLXWp9-jzw6V6XlgfVlO

Experiment 018: tiếp tục exp016 với test số 1

https://colab.research.google.com/drive/1flZh8YnzLuRnqcn0vgLT-6dMjztH995c

Experiment 019: tiếp tục exp016 với full data

https://colab.research.google.com/drive/18nOEOMtW3LxiKh0r9GKpHdUomzRLgx81#scrollTo=Rb7KoY9Mi4Oj

Experiment 020: thí nghiệm 9 mixtures, test số 1

https://colab.research.google.com/drive/1F1b0T7HuR9lMTtgRu9mbB_wUpX6TZu6q

Experiment 021: tiếp tục exp020 với test số 3

https://colab.research.google.com/drive/184JBrUwxyRt_XXfR1gL2Zbg34C2_IG1a#scrollTo=PI0EhE5DkWmT

Experiment 022: tiếp tục exp020 với test số 4 (sử dụng weight từ exp021) → Failed!

https://colab.research.google.com/drive/18-eLmk-R7DnsTuHPbUwBPXLjANpBvd_S#scrollTo=eiZH8JqZi4VE

Experiment 023: tiếp tục exp010 với full data → Failed!

https://colab.research.google.com/drive/1w3e37OXPeguu4G3XCuUtePxkQ4pQTV7z

Tham khảo

[1]. D. Jurafsky, J.H. Martin (2020). *Speech and Language Processing (3rd ed. draf).*

Truy cập tại: https://web.stanford.edu/~jurafsky/slp3/

[2]. L. R. Rabiner, J. G. Wilpon and F. K. Soong, *"High performance connected digit recognition using hidden Markov models,"* in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 8, pp. 1214-1225, Aug. 1989, doi: 10.1109/29.31269.

[3]. J. Picone, *"Continuous speech recognition using hidden Markov models,"* in *IEEE ASSP Magazine*, vol. 7, no. 3, pp. 26-41, July 1990, doi: 10.1109/53.54527.

Truy cập tại: https://www.isip.piconepress.com/publications/journals_refereed/1990/ieee_asspm/hmms/paper_v00.pdf

[4]. N.L.Huy (2019). *Sơ lược về Mel Frequency Cepstral Coefficients (MFCCs)* trên Viblo

Truy cập tại: https://viblo.asia/p/so-luoc-ve-mel-frequency-cepstral-coefficients-mfccs-1VgZv1m2KAw

[5]. Haytham Fayek (2016), *Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between*, online blog

Truy cập tại: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html#fnref:1

[6]. J. Hui (2019), *Speech Recognition — GMM, HMM* on *Medium*

Truy cập tại: https://jonathan-hui.medium.com/speech-recognition-gmm-hmm-8bb5eff8b196