ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Học Máy

Báo cáo Bài Tập Lớn

"Nhận dạng chữ số qua giọng nói sử dụng Mô hình Hidden Markov"

Giảng viên hướng dẫn: TS. Lê Thành Sách

Sinh viên thực hiện:

Lê Chí Đại	2310621 *(Nhóm trưởng)*
Nguyễn Quốc Huy	2311209
Phạm Lê Tiến Đạt	2310687
Võ Văn Thịnh	2313318

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 10 NĂM 2025

Mục lục

Danh sách các ảnh

Danh sách các bảng

Danh sách mã nguồn

Bảng phân công công việc

STT	Họ tên	MSSV	Nhiệm vụ	Đánh giá
1	Lê Chí Đại	2310621	EDA, trích xuất đặc trưng, demo gradio.	100%
2	Nguyễn Quốc Huy	2311209	Thực hiện Bài tập 2	100%
3	Phạm Lê Tiến Đạt	2310687	Thực hiện Bài tập 3	100%
4	Võ Văn Thịnh	2313318	Xây dựng và cài đặt mô hình HMM.	100%

Bảng 1: Bảng phân công công việc

1 Giới thiệu

1.1 Bài toán và Mục tiêu

Nhận dạng giọng nói (Speech Recognition) là một trong những lĩnh vực trọng tâm của Trí tuệ nhân tạo và Xử lý tín hiệu số, hướng tới việc cho phép máy tính hiểu và phản hồi lại lời nói của con người. Trong đó, **nhận dạng chữ số qua giọng nói** là một bài toán cơ bản, nhưng lại mang ý nghĩa nền tảng và có tính ứng dụng thực tiễn cao, ví dụ như trong các hệ thống nhập liệu mã PIN, xác thực danh tính qua số điện thoại, hoặc các ứng dụng ngân hàng tự động.

Tuy nhiên, tín hiệu tiếng nói là một dạng dữ liệu phức tạp, mang tính tuần tự theo thời gian và biến thiên. Tín hiệu này thay đổi đáng kể tùy thuộc vào người nói (giọng, tốc độ, âm sắc) và môi trường (nhiễu nền). Điều này đặt ra thách thức về việc xây dựng một mô hình đủ mạnh mẽ để nắm bắt được bản chất của tín hiệu nhưng cũng đủ linh hoạt để thích ứng với các biến thể.

Mô hình Hidden Markov (HMM) là một lựa chọn kinh điển và hiệu quả cho bài toán này. HMM là một mô hình thống kê cho phép mô hình hóa các chuỗi dữ liệu có trật tự theo thời gian. Thay vì xử lý tín hiệu một cách tĩnh, HMM xem tiếng nói như một quá trình sinh ra từ một chuỗi các ”trạng thái ẩn” (có thể tương ứng với các âm vị hoặc bán âm vị).

Từ đó, báo cáo này đặt ra các mục tiêu cụ thể sau:

- **Nghiên cứu lý thuyết**: Nắm được cơ sở lý thuyết về xử lý tín hiệu âm thanh, đặc biệt là phương pháp trích xuất đặc trưng Mel-Frequency Cepstral Coefficients (MFCCs), kiến trúc của mô hình Hidden Markov (HMM) và cách áp dụng các lý thuyết đó vào thực tế.
- **Xây dựng mô hình và pipeline**: Thiết kế và triển khai mô hình HMM hoàn chỉnh từ đầu (scratch) có khả năng nhận dạng 10 chữ số (từ 0 đến 9) từ các âm thanh đầu vào và xây dựng một pipeline hoàn chỉnh từ tiền xử lý, trích xuất đặc trưng (MFCC) đến huấn luyện và đánh giá..
- **Huấn luyện và Đánh giá**: Huấn luyện mười mô hình HMM riêng biệt, mỗi mô hình cho một chữ số (từ 0 đến 9), trên các đặc trưng MFCC đã trích xuất, và đánh giá hiệu suất của mô hình một cách khách quan thông qua các độ đo.
- **Demo Sản phẩm**: Xây dựng một giao diện web tương tác bằng Gradio, cho phép dự đoán trong thời gian thực với các mẫu âm thanh mới.

1.2 Cấu trúc Báo cáo

Báo cáo được chia thành 4 phần chính, với nội dung như sau:

- **Phần 1 - Giới thiệu**: Trình bày tổng quan về bài toán nhận dạng chữ số qua giọng nói, mục tiêu nghiên cứu và cấu trúc báo cáo.
- **Phần 2 - Cơ sở lý thuyết và Phương pháp**: Trình bày dữ liệu để xây dựng mô hình, các khái niệm và kỹ thuật nền tảng như trích xuất đặc trưng MFCCs, mô hình Hidden Markov (HMM), cũng như phương pháp hiện thực mô hình HMM cho bài toán nhận dạng chữ số qua giọng nói.
- **Phần 3 - Thí nghiệm và Kết quả:** Mô tả quá trình thiết lập thí nghiệm, đánh giá hiệu suất của mô hình, phân tích các kết quả thu được và Demo sản phẩm.
- **Phần 4 – Kết luận và Hướng phát triển**: Tổng kết những kết quả đạt được, nêu ra các hạn chế ở mô hình hiện tại, và đề xuất những hướng nghiên cứu, cải tiến tiếp theo nhằm nâng cao độ chính xác và khả năng ứng dụng của mô hình hiện tại.

2 Cơ sở Lý thuyết và Phương pháp

2.1 Dữ liệu

Hình 1: Ảnh minh họa dataset

Nhóm đã sử dụng bộ dữ liệu âm thanh **Free Spoken Digit Dataset (FSDD)**. Đây là một bộ dữ liệu mở (open dataset), thường được xem như ”bộ MNIST cho âm thanh” được lấy trên nền tảng Kaggle. Nó bao gồm các bản ghi chữ số được nói trong tệp wav ở tần số 8kHz. Các bản ghi âm được cắt bớt để chúng có khoảng lặng gần như tối thiểu ở phần đầu và phần cuối.

Đặc tả bộ dữ liệu:

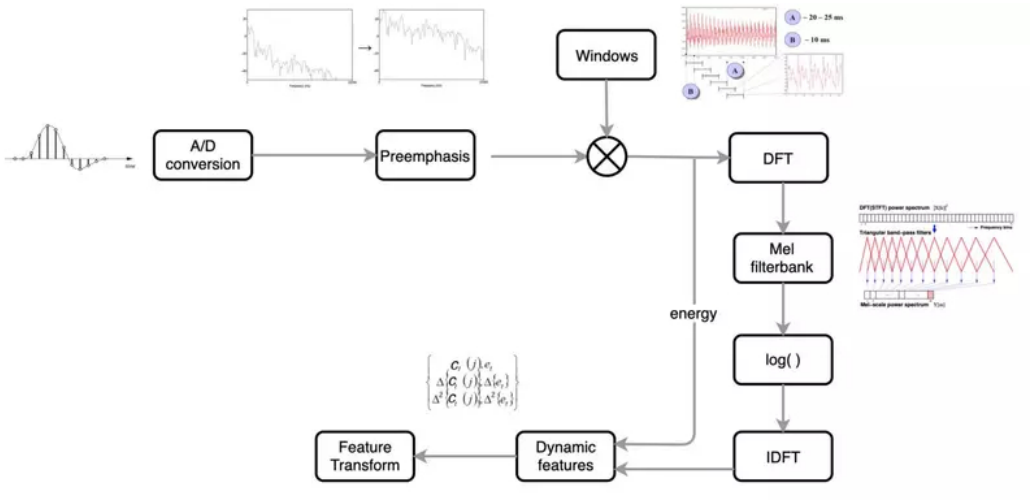
- **Nội dung**: Bộ dữ liệu bao gồm 3.000 bản ghi âm các chữ số (từ 0 đến 9) được phát âm bằng tiếng Anh.
- **Nguồn thu thập**: Dữ liệu được thu từ 6 người nói khác nhau.
- **Phân bố**: Dữ liệu được phân bố đồng đều, mỗi người nói đóng góp 50 bản ghi cho mỗi chữ số
- **Định dạng**: Tất cả các tệp đều ở định dạng .wav với tần số lấy mẫu là 8kHz.

2.2 Trích xuất Đặc trưng: MFCCs

Trong bất kỳ bài toán nhận dạng tiếng nói nào, tín hiệu âm thanh thô (dạng sóng - waveform) ban đầu chứa một lượng thông tin rất lớn, phức tạp, có độ nhiễu cao và mang tính thừa thãi. Nếu đưa trực tiếp tín hiệu thô vào mô hình sẽ không hiệu quả. Mục tiêu của bước trích xuất đặc trưng là chuyển đổi tín hiệu âm thanh thành một dạng biểu diễn cô đọng, ổn định, chỉ giữ lại những thông tin cốt lõi nhất liên quan đến nội dung (content) của lời nói, đồng thời loại bỏ các thông tin không liên quan như nhiễu nền, tông giọng (pitch) hay đặc điểm sinh lý riêng của người nói.

Trong bài tập lớn này, nhóm sử dụng **Mel frequency cepstral coefficients (MFCCs)**, một kỹ thuật trích xuất đặc trưng ”tiêu chuẩn vàng” (gold standard) trong nhận dạng tiếng nói, đặc biệt là khi kết hợp với Mô hình Markov Ẩn (HMM). Lý do chính MFCC hoạt động hiệu quả là vì nó được thiết kế dựa trên mô phỏng cách thức tai người cảm nhận âm thanh, tập trung vào các dải tần số thấp, nơi chứa nhiều thông tin quan trọng nhất của tiếng nói.

Quy trình trích xuất MFCC là một chuỗi các bước xử lý tín hiệu phức tạp, được tóm tắt trong sơ đồ tổng quan tại Hình 2.



Hình 2: Sơ đồ khối tổng quan các bước tính toán MFCC và đặc trưng động.

Quy trình xử lý đặc trưng cho tín hiệu tiếng nói được thực hiện qua các giai đoạn chính như sau (theo Hình 2):

1. **Giai đoạn Tiền xử lý (Preprocessing):** Tín hiệu âm thanh thô được chuyển đổi A/D (Analog-to-Digital), sau đó qua bộ lọc **Pre-emphasis** và được chia thành các khung ngắn (Frames) bằng cách sử dụng các hàm **Windows** (Cửa sổ).

2. **Giai đoạn Chuyển đổi Tần số:** Mỗi khung sau đó được áp dụng **DFT** (Discrete Fourier Transform) để chuyển sang miền tần số, tạo ra phổ năng lượng.

3. **Giai đoạn Lọc Đặc trưng:** Phổ năng lượng này được đưa qua **Mel Filterbank** (Bộ lọc Thang Mel) và áp dụng hàm **Logarithm** (log()). Các bước này mô phỏng cách tai người cảm nhận âm thanh.

4. **Giai đoạn Nén và Trích xuất Đặc trưng Tĩnh:** Kết quả log-energy được áp dụng **IDFT** (thường là DCT trong ngữ cảnh này) để nén thông tin, tạo ra các hệ số MFCC tĩnh (c).

5. **Giai đoạn Trích xuất Đặc trưng Động (Dynamic Features):** Để cung cấp thông tin về sự thay đổi của tiếng nói theo thời gian, các hệ số tĩnh được tính toán đạo hàm bậc nhất (Delta) và bậc hai (Delta-Delta), tổng hợp thành vector đặc trưng 39 chiều. Vector này sẵn sàng để đưa vào Mô hình Markov Ẩn (HMM).

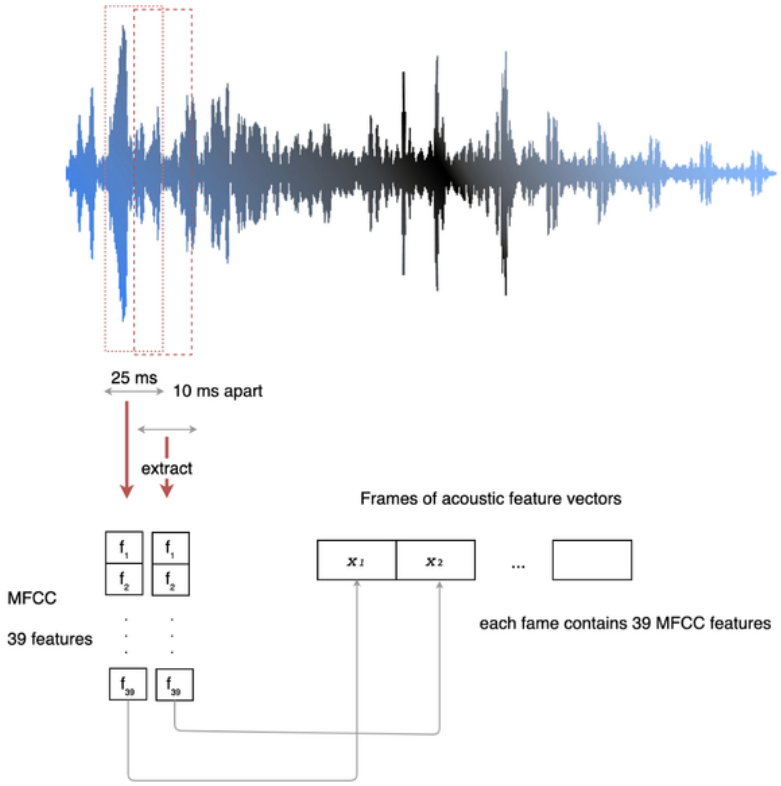
Các bước tính toán cụ thể được thực hiện như sau:

2.2.1 Phân đoạn (Framing) và Cửa sổ (Windowing)

Tín hiệu tiếng nói về bản chất là phi dừng (non-stationary), tức là các đặc tính của nó thay đổi liên tục theo thời gian. Tuy nhiên, nếu ta xét trong một khoảng thời gian rất ngắn (khoảng 20-30ms), tín hiệu có thể được xem là ”gần-dừng” (quasi-stationary).

Do đó, tín hiệu âm thanh liên tục được chia thành các khung (frames) ngắn. Các khung này chồng lấn (overlap) lên nhau để đảm bảo sự liên tục và không mất mát thông tin ở các điểm nối. Trong project này, mỗi khung có độ dài 25ms và bước nhảy (hop) giữa các khung là 10ms (Hình 3).

Một hàm cửa sổ (ví dụ: Hamming) cũng được áp dụng cho từng khung để làm mượt tín hiệu ở hai đầu, giảm thiểu hiện tượng ”rò rỉ phổ” (spectral leakage) khi thực hiện phép biến đổi Fourier ở bước tiếp theo.



Hình 3: Quá trình phân đoạn tín hiệu: Tín hiệu được chia thành các khung 25ms, chồng lấn lên nhau với bước nhảy 10ms. Mỗi khung sau đó được chuyển đổi thành một vector 39 đặc trưng.

2.2.2 Pre-emphasis và DFT

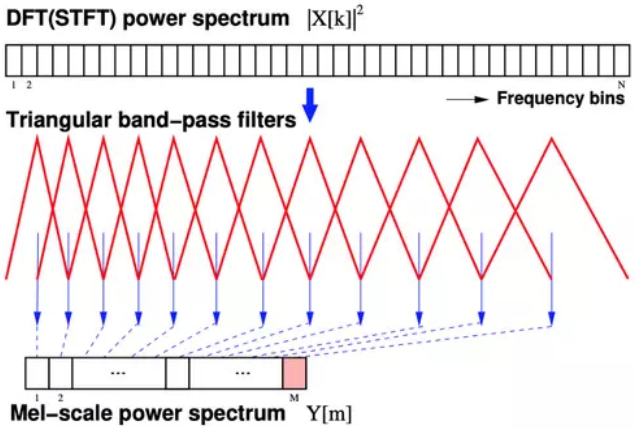
Sau khi qua cửa sổ, một bộ lọc pre-emphasis thường được áp dụng để tăng cường năng lượng cho các thành phần tần số cao, vốn thường bị suy yếu trong quá trình phát âm nhưng lại quan trọng để phân biệt các phụ âm.

Tiếp theo, phép **Biến đổi Fourier Rời rạc (DFT)**, thường được tính toán hiệu quả bằng thuật toán Fast Fourier Transform (FFT), được áp dụng cho từng khung. Bước này chuyển đổi tín hiệu từ miền thời gian (time-domain) sang miền tần số (frequency-domain), cho phép ta phân tích phổ (spectrum) của âm thanh trong khung đó. Kết quả của bước này là Phổ công suất (Power Spectrum) $|X[k]|^2$.

2.2.3 Bộ lọc Thang Mel (Mel Filterbank)

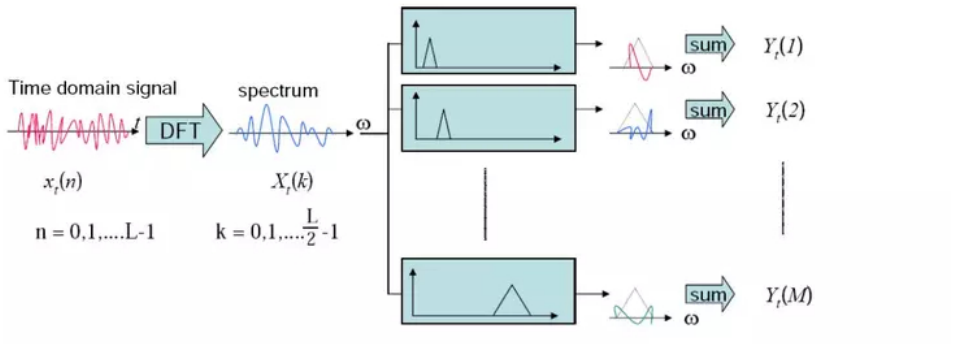
Đây là bước cốt lõi của MFCC. Thay vì phân tích phổ tần số tuyến tính (như trong DFT), ta phân tích phổ theo thang Mel (Mel-scale), vốn mô phỏng độ nhạy của tai người. Tai người nhạy cảm hơn với sự thay đổi ở các tần số thấp (dưới 1000Hz) và kém nhạy cảm hơn ở các tần số cao.

Một tập hợp các bộ lọc thông dải hình tam giác (Triangular band-pass filters) được tạo ra. Các bộ lọc này được đặt cách đều nhau theo thang Mel (tương đương với việc chúng nằm dày đặc ở tần số thấp và thưa dần ở tần số cao trên thang tuyến tính).



Hình 4: Áp dụng bộ lọc tam giác trên thang Mel: Phổ công suất DFT (Frequency bins) được nhân với các bộ lọc tam giác để tính tổng năng lượng cho mỗi dải tần Mel.

Như minh họa trong Hình 4 và 5, phổ công suất $|X[k]|^2$ từ bước DFT được truyền qua bộ lọc này. Năng lượng trong mỗi bộ lọc được tính bằng cách nhân và lấy tổng (sum), tạo ra một vector M giá trị năng lượng (với M là số lượng bộ lọc, thường là 20-40).



Hình 5: Minh họa quá trình lọc: Phổ tín hiệu (spectrum) được đưa qua các bộ lọc (filterbank) khác nhau, sau đó lấy tổng năng lượng để ra các hệ số $Y(M)$.

2.2.4 Logarithm và Biến đổi Cosine Rời rạc (DCT)

Tai người cũng cảm nhận âm lượng theo thang logarit. Do đó, chúng ta lấy logarit của các giá trị năng lượng $Y[m]$ đã tính ở bước trên. Bước này giúp nén dải động (dynamic range) và làm cho đặc trưng ít bị ảnh hưởng bởi sự thay đổi về âm lượng.

Cuối cùng, phép **Biến đổi Cosine Rời rạc (DCT)** được áp dụng lên M giá trị log-energy này. DCT là một phép biến đổi ”thực” của Fourier, có tác dụng ”giải tương quan” (de-correlate) các đặc trưng và nén phần lớn thông tin năng lượng vào các hệ số đầu tiên. Kết quả của DCT chính là các Hệ số Cepstral (Cepstral Coefficients).

Thông thường, ta chỉ giữ lại 13 hệ số đầu tiên (bao gồm hệ số c_0, đại diện cho năng lượng tổng của khung, và 12 hệ số c_1 đến c_{12} mô tả hình dạng phổ).

2.2.5 Đặc trưng Động (Dynamic Features)

13 hệ số MFCC vừa tính được gọi là đặc trưng "tĩnh" (static), vì chúng chỉ mô tả hình dạng phổ của một khung 25ms. Tuy nhiên, tiếng nói là một quá trình động (dynamic), sự thay đổi của phổ theo thời gian mới là yếu tố then chốt để nhận dạng.

Để cung cấp thông tin về sự thay đổi này cho HMM, ta tính toán thêm các đặc trưng động:

- **Deltas (Delta):** Đạo hàm bậc nhất (vận tốc) của các hệ số tĩnh, cho biết chúng đang tăng hay giảm.
- **Delta-Deltas (Delta-Delta):** Đạo hàm bậc hai (gia tốc) của các hệ số tĩnh, cho biết tốc độ thay đổi của Deltas.

Bằng cách kết hợp 13 hệ số tĩnh, 13 hệ số delta, và 13 hệ số delta-delta, ta thu được một **vector đặc trưng 39 chiều** cho mỗi khung 10ms (như minh họa ở cuối Hình 3).

2.2.6 Hiện thực và Trực quan hóa Đặc trưng

Trong quá trình hiện thực, nhóm đã sử dụng thư viện **librosa** để thực hiện toàn bộ quy trình trích xuất đặc trưng.

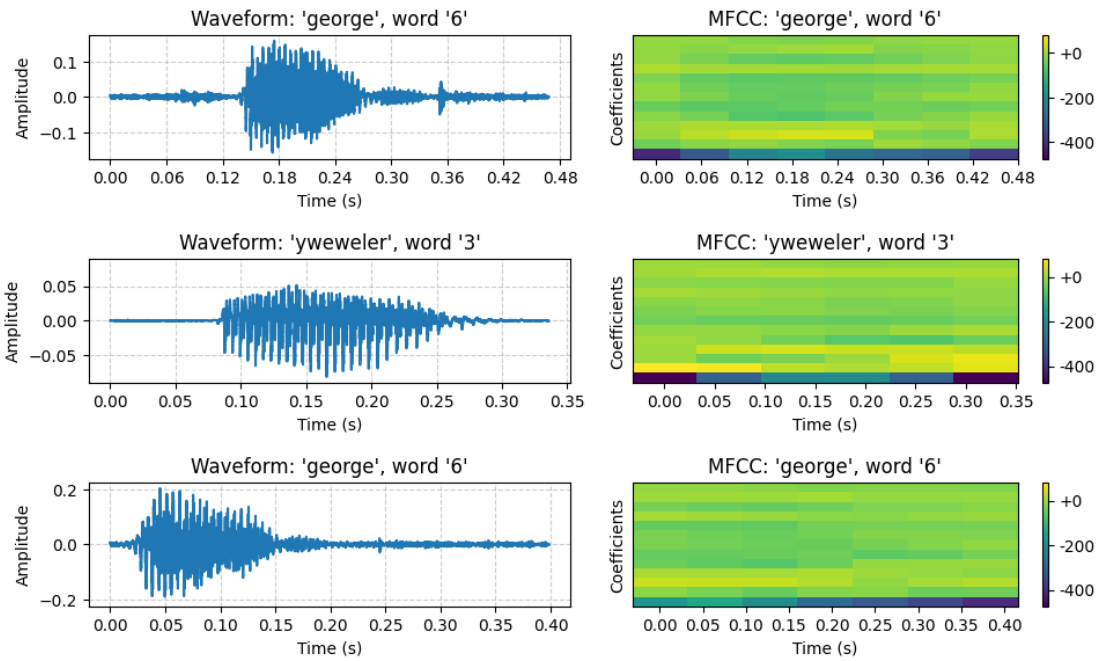
Lựa chọn Đặc trưng: Để đơn giản hóa mô hình hóa chuỗi ban đầu và tập trung vào hiệu suất của các hệ số phổ cơ bản, nhóm đã chọn sử dụng **13 hệ số MFCC tĩnh (`n_mfcc=13`)** làm vector đặc trưng chính cho mỗi khung thời gian. Đây là bước đầu tiên và cơ bản nhất trong việc xây dựng hệ thống.

Đoạn mã Python dưới đây thể hiện việc sử dụng hàm chính của thư viện `librosa` để trích xuất 13 hệ số MFCC tĩnh và trực quan hóa kết quả:

```
mfccs = [librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13) for y
    in ys]
```

Mã nguồn 1: Mã nguồn trích xuất MFCC tĩnh

Kết quả của quá trình này là biến đổi mỗi tệp âm thanh thô thành một chuỗi (sequence) các vector đặc trưng **13 chiều tĩnh**. Hình 6 cho thấy kết quả trực quan hóa của các hệ số MFCC này so với dạng sóng ban đầu. Chuỗi đặc trưng 13 chiều này chính là đầu vào cho quá trình huấn luyện mô hình HMM. Các nghiên cứu mở rộng trong tương lai có thể xem xét bổ sung các đặc trưng động (Delta và Delta-Delta) để đạt được hiệu suất nhận dạng tối ưu hơn.



Hình 6: Kết quả trực quan hóa MFCC: (Trái) Dạng sóng âm thanh thô trong miền thời gian. (Phải) Biểu diễn MFCC tĩnh (13 hệ số) trong miền thời-tần, trục hoành là thời gian (khung), trục tung là các hệ số, màu sắc biểu thị giá trị của hệ số.

2.3 Mô hình Markov Ẩn (HMM)

2.3.1 Định nghĩa

Mô hình Markov Ẩn - *Hidden Markov Model* là một mô hình xác suất dùng để mô tả các hệ thống có trạng thái ẩn (không thể quan sát trực tiếp), nhưng có thể suy ra gián tiếp từ các quan sát đầu ra. Một HMM bao gồm ba thành phần chính:

- **Phân phối trạng thái khởi đầu** - π: là một phân phối xác suất của trạng thái khởi đầu cho chuỗi quan sát.
- **Phân phối chuyển đổi trạng thái** - **A**: là phân phối xác suất chuyển từ trạng thái này sáng trạng thái kia của hệ thống.
- **Phân phối của quan sát** - **B**: là phân phối xác suất của quan sát (rời rạc hoặc liên tục) tại các trạng thái của hệ thống.

2.3.2 Các bài toán cơ bản của HMM

Một mô hình HMM có tham số là $\lambda = (\pi, \mathbf{A}, \mathbf{B})$ gồm N trạng thái, gọi:

- π_i: xác suất khởi động từ trạng thái i, với $i = 1, 2, ...N$.
- a_{ij}: xác suất chuyển từ trạng thái i sang j, với $i, j = 1, 2, ...N$.
- $b_i(o)$: xác suất quan sát được o tại trạng thái i.

Xét chuỗi quan sát thu được sau T trạng thái là $O = O_1O_2O_3...O_T$. Ta có các bài toán chính sử dụng HMM như sau.

Bài toán đánh giá (Evaluation)

Mục tiêu bài toán là tính xác suất xảy ra chuỗi quan sát đó. Cụ thể, ta cần tính:

$$P(O|\lambda) = P(O_1, O_2, O_3, ..., O_T|\lambda) \tag{1}$$

Giải thuật *Forward* được dùng trong bài toán này để tìm ra xác suất xảy ra của chuỗi quan sát. Ý tưởng của giải thuật là tính tổng xác suất tất cả con đường có thể để có được chuỗi quan sát đó.

- Giả sử ta đã có xác suất xảy ra chuỗi quan sát $O_1O_2...O_{t-1}$ kết thúc tại trạng thái i tại thời điểm $t-1$ là $\alpha_{t-1}(i)$.
- Khi đó, xác suất xảy ra chuỗi $O_1O_2...O_t$ kết thúc tại trạng thái i tại thời điểm là $\alpha_t(i)$ sẽ được tính tổng trên N con đường từ quan sát trước đó.

$$\alpha_t(i) = \sum_{k=1}^{N} [\alpha_{t-1}(k)a_{ki}]b_i(O_t) \tag{2}$$

  Tiếp tục thực hiện cho đến khi hoàn thành chuỗi quan sát.

- Xác suất của quan sát đầu tiên tại trạng thái i được tính là $\alpha_1(i) = \pi_i b_i(O_1)$.
- Cuối cùng, xác suất cần tìm sẽ là tổng xác suất xảy ra chuỗi đầy đủ kết thúc tại thời điểm

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{3}$$

Bài toán giải mã (Decoding)

Bài toán này sẽ giúp tìm ra chuỗi trạng thái có khả năng sinh ra chuỗi quan sát O nhất. Giải thuật *viterbi* được dùng cho bài toán này có ý tưởng như sau:

- Giả sử đã biết chuỗi trạng thái tối ưu $q_1^* q_2^* ... q_{t-1}^*$ sinh ra ra chuỗi quan sát $O_1 O_2 ... O_{t-1}$, và kết thúc tại trạng thái i. Xác suất của con đường tối ưu này là $\delta_{t-1}(i)$.
- Bằng phương pháp quy nạp, ta sẽ tính:

$$\delta_t(i) = \max_{1 \leq k \leq N} \left[\delta_{t-1}(k) a_{ki} \right] b_i(O_t) \tag{4}$$

$$\varphi_t(i) = \arg\max_{1 \leq k \leq N} \left[\delta_{t-1}(k) a_{ki} \right] \tag{5}$$

  Trong đó $\varphi_t(i)$ dùng để truy vết lại con đường tối ưu sau này.

- Khởi động giá trị của δ và φ:

$$\delta_1(i) = \pi_i b_i(O_1) \tag{6}$$

$$\varphi_1(i) = 0 \tag{7}$$

- Sau khi tìm được xác suất quan sát được O tối ưu kết thúc tại từng trạng thái, ta chọn ra trường hợp có xác suất cao nhất.

$$P^* = \max_{1 \leq k \leq N} \delta_T(k) \tag{8}$$

$$q_T^* = \arg\max_{1 \leq k \leq N} \delta_T(k) \tag{9}$$

- Chuỗi trạng thái tối ưu được truy vết lại:

$$q_t^* = \varphi_{t+1}(q_{t+1}^*) \tag{10}$$

Bài toán học (Learning)

Để tìm ra được mô hình HMM có bộ tham số $\lambda = (\pi, \mathbf{A}, \mathbf{B})$ mô tả tốt hệ thống trạng thái ẩn, ta cần tính toán các thành phần sau:

- Giải thuật lan truyền ngược (Backward Algorithm):
  - Khởi động: tính $\beta_T(i) = 1$ với $\forall i = 1, 2, ...N$
  - Tính lần lượt $\beta_t(i)$ với $t = T-1, T-2, ...1$ theo công thức sau:

$$\beta_t(i) = \sum_{k=1}^{N} \Big[a_{ik} b_k(O_{t+1}) \beta_{t+1}(k) \Big] \tag{11}$$

- Xác suất trạng thái tại thời điểm t là i khi biết mô hình HMM và chuỗi quan sát:

$$\gamma_t(i) = P(q_t = S_i | O, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \tag{12}$$

$$= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{13}$$

- Xác suất trạng thái tại thời điểm t là i và $t+1$ là j.

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda) \tag{14}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \tag{15}$$

Giải thuật Baum-Welch để tối ưu tham số gồm hai bước chính là E-step và M-step:

- Khởi động: Khởi tạo ngẫu nhiên hoặc bằng các phương pháp như K-means các ma trận $\pi, \mathbf{A}, \mathbf{B}$. Lặp lại E-step và M-step cho đến khi hội tụ.
- Thực hiện tính kỳ vọng (E-step)
  - Tính $\alpha_t(i)$ và $\beta_t(i)$ dùng giải thuật lan truyền thuận và lan truyền ngược.
  - Tính $\xi_t(i)$ và $\gamma_t(i)$ theo công thức 13 và 15.
- Thực hiện bước cực đại hóa kỳ vọng (M-step) để tính $\pi, \mathbf{A}, \mathbf{B}$:

2.4 Hiện thực mô hình HMM cho bài toán nhận diện chữ số

2.4.1 Mô hình bài toán theo HMM

Từ ma trận đặc trưng MFCC của mỗi audio, ta sẽ xây dựng một mô hình HMM cho mỗi chữ số (label) gồm N trạng thái. Quan sát của mỗi trạng thái là vector $v \in \mathbb{R}^F$. Giả sử v tuân theo một phân phối Gaussian đa biến $v \sim \mathcal{N}(\mu, \Sigma)$. Vì vậy ta sẽ xây dựng Gaussian HMM cho từng chữ số và huấn luyện để tìm ra bộ tham số cho từng mô hình.

2.4.2 Hiện thực mô hình Gaussian HMM

Ta sẽ xây dựng một class `GaussianHMM` kế thừa lớp `BaseEstimator` của thư viện sklearn.

```
class GaussianHMM(BaseEstimator):
    def __init__(self,
        n_components=5, #num hidden states
        n_iter=100, #num optimizing iteration
        covariance_type='diag', #only for diag
        tol=1e-4, #for convergence check
        verbose=False): #log training info
```

Khởi tạo tham số

Tương tự như mô hình HMM thông thường, Gaussian HMM cũng bao gồm bộ tham số $\lambda = (\pi, \mathbf{A}, \mathbf{B})$ và được khởi tạo mặc định phân phối đều ở ma trận π và $\mathbf{A}$. Còn mean và covariance cho các phân phối Gaussian tại các trạng thái có thể khởi tạo ngẫu nhiên. Tuy nhiên, nhóm hiện thực phương pháp K-means Clustering, cho phép khởi tạo tốt hơn bằng cách gom cụm dữ liệu thành N cụm, sau đó lấy các điểm trung tâm từng cụm là mean cho từng trạng thái.

```
def _initialize_parameters(self, X):
    """Better initialization using K-means"""
    n_samples, n_features = X.shape

    # Use K-means for better initialization
    if n_samples >= self.n_components:
        kmeans = KMeans(n_clusters=self.n_components,
  random_state=42, n_init=10)
        labels = kmeans.fit_predict(X)
```

```
        self.means_ = kmeans.cluster_centers_
        self.covars_ = np.zeros((self.n_components,
n_features))

        for i in range(self.n_components):
            mask = labels == i
            if np.sum(mask) > 0:
                self.covars_[i] = np.var(X[mask], axis=0)
+ EPS
            else:
                self.covars_[i] = np.var(X, axis=0) + EPS
     else:
        # Fallback: add random perturbations
        mean = np.mean(X, axis=0)
        var = np.var(X, axis=0) + EPS

        self.means_ = mean + np.random.randn(self.
n_components, n_features) * np.sqrt(var) * 0.1
        self.covars_ = np.tile(var, (self.n_components, 1)
)

     # Initialize log probabilities
     self.log_pi = np.log(self.pi + EPS)
     self.log_A = np.log(self.A + EPS)
```

Huấn luyện Gaussian HMM

Bằng việc lặp lại giải thuật tối ưu Baum-Welch nhiều lần cho đến khi hội tụ, ta sẽ thu được mô hình HMM mô tả tốt cho hệ thống trạng thái ẩn của tiếng đọc chữ số đó. Khác một chút với HMM cơ bản, Gaussian HMM sẽ tìm ra phân phối Gaussian của quan sát tại từng trạng thái, theo hai công thức sau:

$$\mu_j^{(new)} = \frac{\sum_{t=1}^{T} \gamma_t(j)\, o_t}{\sum_{t=1}^{T} \gamma_t(j)} \tag{16}$$

$$\Sigma_j^{(new)} = \frac{\sum_{t=1}^{T} \gamma_t(j)\, (o_t - \mu_j^{(new)})(o_t - \mu_j^{(new)})^\top}{\sum_{t=1}^{T} \gamma_t(j)} \tag{17}$$

Để tránh bị mất ổn định số học do tích các xác suất qua nhiều trạng thái khiến nó gần về 0, ta sẽ hiện thực tất cả trong không gian log.

Dự đoán bằng Gaussian HMM

Sau khi tìm được tập các tham số của mô hình HMM tương ứng cho từng chữ số, ta sẽ dự đoán bằng cách tìm ra mô hình HMM có xác suất sinh ra chuỗi quan sát là lớn nhất.

```
def score(self, X, lengths=None):
    """
    Compute the log probability under the model using
   Forward algorithm.
    """
    if lengths is None:
        lengths = [X.shape[0]]

    log_prob = 0
    pos = 0

    for length in lengths:
        obs_seq = X[pos:pos + length]
        log_B = self._compute_log_emission_prob(obs_seq)

        # Use FORWARD algorithm to compute log probability
        log_alpha = self._forward_log(log_B)
        log_prob += logsumexp(log_alpha[:, -1])  # CORRECT
   : sum over all paths

        pos += length

    return log_prob
```

3 Thí nghiệm và Kết quả

3.1 Thiết lập Thí nghiệm

Các thí nghiệm được thực hiện trong môi trường Python 3.13.5, sử dụng các thư viện chính bao gồm:

- **NumPy**, **SciPy** cho xử lý tín hiệu và toán học.
- **librosa** cho trích xuất đặc trưng âm thanh (MFCC, năng lượng, ZCR).
- **matplotlib**, **seaborn** cho trực quan hoá dữ liệu.
- **Gradio** cho xây dựng giao diện demo tương tác.

Bộ dữ liệu được sử dụng là **Free Spoken Digit Dataset (FSDD)**, gồm các mẫu âm thanh của các chữ số từ `0–9`, được phát âm bởi nhiều người nói khác nhau. Dữ liệu được chia theo tỉ lệ **70% train** và **30% test**, đảm bảo không có sự chồng lấn giữa hai tập.

Tham số trích xuất đặc trưng:

- Số hệ số MFCC: 13
- Kích thước cửa sổ: 32ms
- Bước nhảy (hop length): 10ms
- Tần số lấy mẫu:8000Hz

Tham số mô hình HMM:

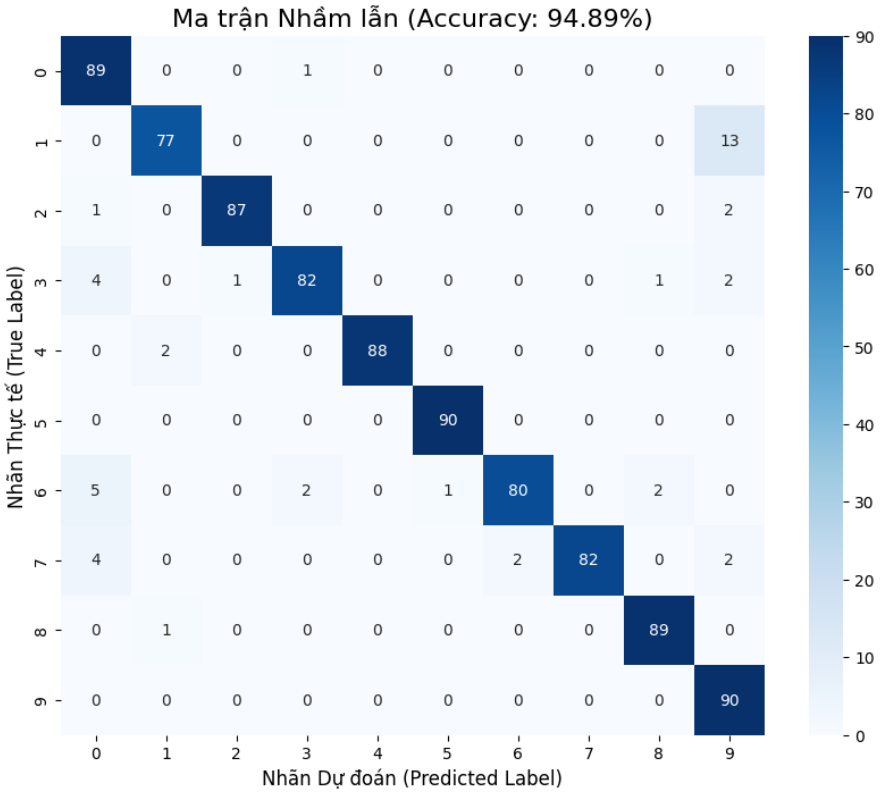
- Số trạng thái ẩn: 5
- Phân phối quan sát: Gaussian.
- Thuật toán huấn luyện: Baum–Welch (Expectation–Maximization).
- Số vòng lặp huấn luyện tối đa: 100

3.2 Kết quả Đánh giá

Độ chính xác trên tập kiểm tra đạt được:

$$\text{Accuracy} = \mathbf{94.89\%}$$

Kết quả được đánh giá dựa trên xác suất likelihood mà mỗi mô hình HMM gán cho chuỗi đặc trưng đầu vào. Mỗi mẫu âm thanh test được chấm điểm trên 10 mô hình tương ứng với các chữ số, và nhãn được chọn là mô hình có xác suất cao nhất.



Hình 7: Ma trận nhầm lẫn thể hiện hiệu suất phân loại giữa các chữ số.

Phân tích ma trận nhầm lẫn cho thấy một số cặp chữ số thường bị nhầm lẫn:

- **“one” vs “nine”**: có phổ âm tương tự nhau ở tần số cao.

3.3 Project page

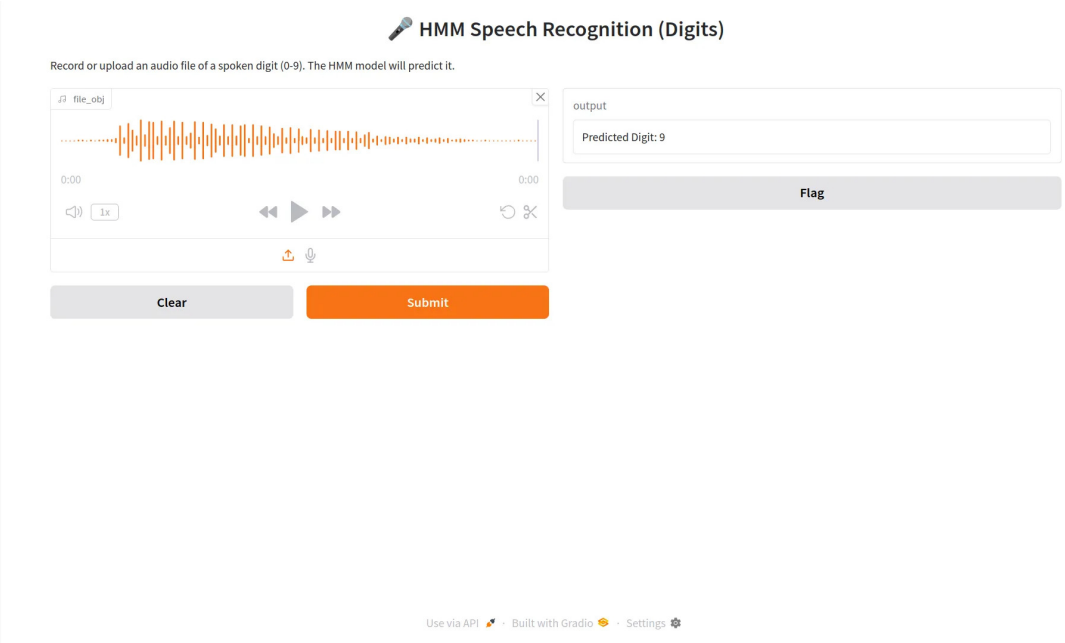
Dự án được trình bày chi tiết kèm hình ảnh, kết quả và mã nguồn tại:

https://caotaytang.github.io/ML-251/

3.4 Demo Sản phẩm

Nhóm đã triển khai một ứng dụng demo tương tác sử dụng **Gradio**, cho phép người dùng:

- Ghi âm trực tiếp bằng micro.
- Xem đặc trưng MFCC tương ứng của âm thanh.
- Hệ thống dự đoán chữ số bằng mô hình HMM huấn luyện sẵn.



Hình 8: Giao diện demo nhận dạng chữ số bằng giọng nói (Gradio).

Ứng dụng này minh họa quy trình ”end-to-end”: từ thu âm → trích xuất MFCC → dự đoán bằng HMM → hiển thị kết quả, giúp trực quan hoá hoạt động của mô hình.

4 Kết luận và Hướng phát triển

4.1 Tổng kết

Nhóm đã xây dựng thành công hệ thống **nhận dạng chữ số qua giọng nói** dựa trên mô hình **Hidden Markov Model (HMM)** và đặc trưng **MFCC**. Mô hình được huấn luyện và đánh giá trên bộ dữ liệu **Free Spoken Digit Dataset (FSDD)**, đạt độ chính xác **94.89%** trên tập kiểm tra. Kết quả này khẳng định tính hiệu quả của việc kết hợp MFCC và HMM trong mô hình hóa tín hiệu tiếng nói tuần tự, đồng thời giúp nhóm hiểu rõ cơ chế toán học của thuật toán Baum–Welch và quy trình huấn luyện HMM from scratch.

4.2 Hạn chế

Một số hạn chế chính của hệ thống bao gồm:

- **Dữ liệu huấn luyện hạn chế**: Bộ FSDD có quy mô nhỏ, chất lượng cao và ít nhiễu, chưa phản ánh thực tế phức tạp của giọng nói tự nhiên.
- **Giả định đơn giản của HMM**: Mô hình giả định các quan sát độc lập theo thời gian, nên chưa mô tả được mối liên hệ dài hạn trong chuỗi âm thanh.
- **Chưa tối ưu tham số**: Quá trình huấn luyện sử dụng các giá trị mặc định, chưa thực hiện tìm kiếm tham số (grid search) để đạt hiệu quả cao nhất.

4.3 Hướng phát triển

Trong tương lai, nhóm dự định mở rộng và cải thiện mô hình theo các hướng sau:

- Sử dụng thêm đặc trưng nâng cao như **MFCC delta** và **delta–delta** để mô tả biến thiên theo thời gian.
- Mở rộng thí nghiệm sang các bộ dữ liệu lớn và đa dạng hơn, chẳng hạn như **Google Speech Commands**.
- Nghiên cứu các hướng kết hợp giữa HMM và học sâu, như:
  - Hệ thống lai **HMM–DNN**.
  - Mạng hồi quy **RNN / LSTM**.
  - Các mô hình **end–to–end** hiện đại (Transformer, Wav2Vec 2.0).

Tài liệu tham khảo

[1] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," Proceedings of the IEEE, vol. 77, no. 2, pp. 257–286, 1989.

[2] D. Jurafsky and J. H. Martin, Speech and Language Processing (3rd ed., draft). Prentice Hall, 2025. Chapter A: Hidden Markov Models, draft of August 24, 2025.

[3] M. Yeluri, "Understanding and implementing speech recognition using hmm." https://maharshi-yeluri.medium.com/understanding-and-implementing-speech-recognition-using-hmm-6a4e7666de1, 2021. Accessed: 2025-11-12.

[4] K. Shariat, "Hidden markov models for speech recognition — supplementary notebook." https://sut-ai.github.io/supplementary/notebooks/hmm_speech_recognition/, 2023. Accessed: 2025-11-12.