
# Handsign Recognition (các phần đã thực hiện ở giữa kì được đánh dấu (*)

Using Machine Learning

Nguyen Vi Chi Cuong  Do Tien Dung  Do Huu Dat
VNU Hanoi University of Science
Ngày 5 tháng 6 năm 2025

# Mục lục

# Giới thiệu bài toán nhận diện chữ cái ký hiệu ASL (*)

**Mục tiêu**: Nhận diện 24 chữ cái trong bảng chữ cái ký hiệu Mỹ (ASL) từ hình ảnh ký hiệu tay.

**Ứng dụng**:

- Hỗ trợ giao tiếp cho người khiếm thính.
- Giáo dục ngôn ngữ ký hiệu.
- Công nghệ trợ giúp.

**Nguồn**: ASL HandSign Dataset trên Kaggle.

**Đặc điểm**:

- Hình ảnh thang độ xám, đã xử lý ngưỡng, có các góc độ khác nhau.
- Bao gồm ký hiệu tay cho 24 chữ cái (trừ J và Z).

# Phương pháp và ý nghĩa(*)

**Mô hình**:

- K-Nearest Neighbors (KNN).
- Naive Bayes.
- Softmax Regression.
- Tiền xử lý: resize, chuẩn hóa, PCA.

**Quy trình**:

1. Tiền xử lý hình ảnh.
2. Huấn luyện mô hình phân loại.
3. Đánh giá hiệu suất.

**Ý nghĩa**

- Cải thiện giao tiếp cho cộng đồng người khiếm thính.
- Ứng dụng trong hệ thống tự động hóa và giáo dục ngôn ngữ ký hiệu.

# Mục lục

# Tiền xử lý dữ liệu: Resize ảnh (*)

**Giới thiệu**: Resize ảnh là bước quan trọng trong tiền xử lý dữ liệu cho bài toán nhận diện ASL.

**Mục đích**:

- Chuẩn hóa kích thước ảnh (ví dụ: 32×32).
- Tạo vector đặc trưng cố định (1024 chiều).
- Cải thiện thời gian huấn luyện của các mô hình học máy.

# Tiền xử lý dữ liệu: Tác động của Resize ảnh(*)

**Chuẩn hóa kích thước**: Đảm bảo tất cả ảnh có cùng kích thước.

**Giảm chiều đặc trưng**: Từ 64×64 (4096) xuống 32×32 (1024), giúp giảm chi phí tính toán.

**Bảo toàn thông tin**: Nội suy tuyến tính kép giữ chi tiết hình dạng tay.

**Cải thiện hiệu suất**: Tăng tốc tính toán của các mô hình học máy, giảm sai số.

# Tiền xử lý dữ liệu: Chuẩn hóa dữ liệu(*)

**Giới thiệu**: Chuẩn hóa dữ liệu là bước quan trọng để cân bằng đặc trưng trong bài toán ASL.

**Mục đích**:

- Đưa dữ liệu về phạm vi giá trị chuẩn.
- Cân bằng ảnh hưởng của các đặc trưng.
- Cải thiện hiệu năng mô hình học máy.

# Chuẩn hóa dữ liệu: Phương pháp Standardization(*)

**Nội dung**: Chuyển dữ liệu về phân bố với trung bình = 0, độ lệch chuẩn = 1.

**Công thức**:

$$X' = \frac{X - \text{mean}(X)}{\text{std}(X)}$$

**Ví dụ**: Với $\text{mean}(X) = 10$, $\text{std}(X) = 4$, $x = 15.9$:

$$x' = \frac{15.9 - 10}{4} = 1.475$$

# Chuẩn hóa dữ liệu: Tác động (*)

**Hội tụ nhanh hơn**: Chuẩn hóa dữ liệu làm cho các đặc trưng có cùng phạm vi giá trị, từ đó Gradient đồng đều, tránh overshooting.

**Cân bằng đặc trưng**:Các đặc trưng có phạm vi giá trị lớn hơn sẽ ảnh hưởng lớn hơn đến hàm mất mát nên khi chuẩn hóa sẽ giúp mọi đặc trưng có cùng phạm vi giá trị.

**Dễ điều chỉnh tham số**: Giảm biến động gradient, ổn định huấn luyện.

**Cải thiện độ chính xác**: Mô hình hóa mối quan hệ dữ liệu hiệu quả hơn.

# Tiền xử lý dữ liệu: Phân tích thành phần chính (PCA)

**Giới thiệu**: PCA giảm chiều dữ liệu trong bài toán ASL.

**Mục đích**:

- Giảm số chiều đặc trưng, loại bỏ nhiễu.
- Giảm chi phí tính toán.
- Cải thiện hiệu năng mô hình học máy trong một số trường hợp.

# PCA: Phương pháp (*)

**Nội dung**: Tìm các hướng có phương sai lớn nhất, chiếu dữ liệu lên không gian mới.

**Các bước**:

- Chuẩn hóa dữ liệu (trung bình = 0, độ lệch chuẩn = 1).
- Tính ma trận hiệp phương sai.
- Phân tích giá trị riêng và vector riêng.
- Chọn $k$ thành phần chính.
- Công thức chiếu dữ liệu: Có dữ liệu gốc $X$ (ma trận n x d với n mẫu và d chiều), ma trận thành phần chính là $W$ (ma trận d x k chứa k vector riêng), dữ liệu sau khi giảm chiều là:

$$X_{\text{PCA}} = X \cdot W$$

**Ví dụ**: Giảm từ 1024 chiều (32×32) xuống 81 chiều, giữ 95% phương sai.

# PCA: Tác động(*)

**Giảm chiều**: Từ 1024 xuống 81, giảm chi phí tính toán.

**Loại bỏ nhiễu**: Giữ thông tin chính về hình dạng tay trong ASL.

**Cải thiện hiệu suất**: Giảm tham số trong Gaussian Naive Bayes, tránh quá khớp.

# Mục lục

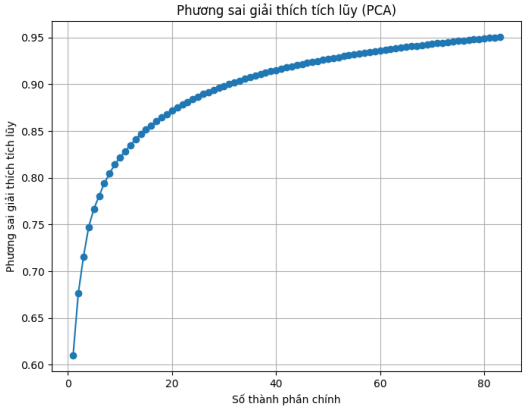
# Bảng Phân Bố Số Lượng Mẫu Theo Nhãn

| Nhãn | Số lượng | Nhãn | Số lượng | Nhãn | Số lượng |
|---|---|---|---|---|---|
| D | 1579 | M | 1082 | P | 931 |
| E | 1421 | Q | 1078 | O | 904 |
| F | 1247 | I | 1069 | | |
| N | 1235 | L | 1047 | | |
| T | 1185 | V | 1026 | | |
| B | 1159 | A | 1023 | | |
| C | 1153 | H | 1022 | | |
| K | 1143 | R | 1021 | | |
| X | 1134 | W | 1008 | | |
| S | 1132 | | | | |
| Y | 1130 | | | | |
| U | 1108 | | | | |
| G | 1096 | | | | |

# Lượng Thông Tin Được Bảo Tồn Theo Phương Sai

**Nhận xét:**

- Phương sai tích lũy tăng nhanh ở các thành phần đầu, hội tụ sau $\sim 40$ PC.
- Với 4 PC: $\sim 74.7\%$ phương sai (gần 75% thông tin).
- Với 6 PC: $\sim 78.1\%$ phương sai (cải thiện nhẹ).
- Chọn 4-6 PC để cân bằng độ phức tạp và thông tin.
- Để giữ $> 90\%$ phương sai, cần 30-40 PC.

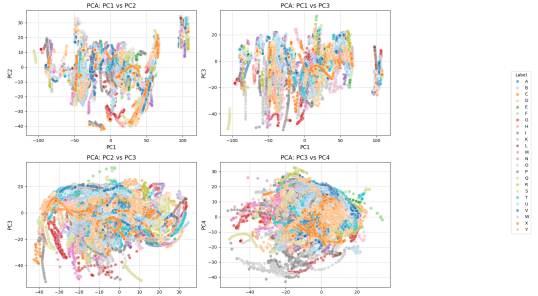



Hình: Biểu đồ phương sai giải thích tích lũy theo số lượng PC

# Trực Quan Hóa Dữ Liệu Theo Cặp Thành Phần Chính

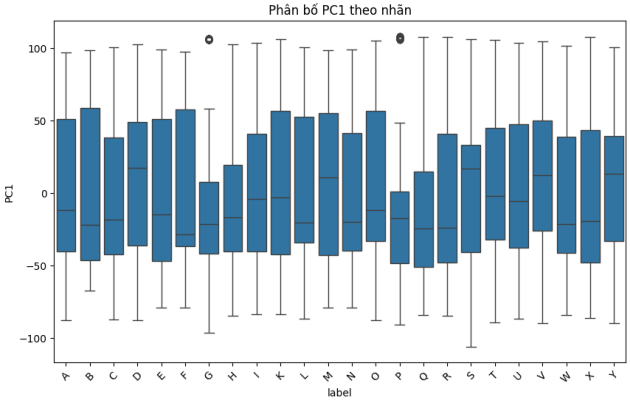
**Nhận xét:**

- Một số cụm nhãn tách biệt rõ, như nhãn **A**, **B**, **D**.
- Nhiều nhãn chồng lấn ở PC1-PC2, PC1-PC3, gây khó khăn cho phân loại.
- Dữ liệu phân bố không đồng đều
- PCA giúp hình dung dữ liệu đa chiều, nhưng chưa tách biệt rõ tất cả nhãn.




Hình: Biểu diễn dữ liệu sau khi giảm chiều PCA theo cặp PC

# Quan Hệ Giữa Các Chiều Dữ Liệu Chính và Đầu Ra




Hình: Phân bố thành phần chính đầu tiên (PC1) theo từng nhãn

# Nhận xét

- PC1 phân bố đồng đều, không tách biệt rõ cụm nhãn, ít thông tin phân biệt.
- Hệ số tương quan Pearson (6 PC đầu):

| Thành phần chính | Tương quan với nhãn |
|---|---|
| PC1 | 0.0017 |
| PC2 | -0.0297 |
| PC3 | 0.0038 |
| PC4 | -0.0178 |
| PC5 | **-0.1507** |
| PC6 | **0.1057** |

- Nhìn chung, các hệ số tương quan đều rất nhỏ, cho thấy các thành phần chính không có mối quan hệ tuyến tính mạnh với nhãn. Tuy nhiên, **PC5 và PC6 có giá trị tương quan tương đối nổi bật** so với các thành phần còn lại.

# Mục lục

# Giới thiệu K-means

**K-means: Thuật toán phân cụm dựa trên trung tâm**

- Mục tiêu: Chia dữ liệu thành $K$ cụm dựa trên khoảng cách Euclidean đến trung tâm cụm.
- Ý tưởng: Tối thiểu hóa tổng bình phương khoảng cách từ các điểm đến trung tâm cụm.
- **Ứng dụng**:
  - Phân đoạn thị trường.
  - Xử lý ảnh (nén, phân đoạn).
  - Phân tích văn bản, nhận diện mẫu.

# Phân tích toán học

**Ký hiệu:**

- Dữ liệu: $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, $\mathbf{x}_i \in \mathbb{R}^d$.
- Trung tâm cụm: $\mathbf{M} = [\mathbf{m}_1, \ldots, \mathbf{m}_K]$.
- Nhãn: $\mathbf{y}_i = [y_{i1}, \ldots, y_{iK}]$, $y_{ij} = 1$ nếu $\mathbf{x}_i$ thuộc cụm $j$, ngược lại $y_{ij} = 0$.
- Ràng buộc: $y_{ij} \in \{0, 1\}, \sum_{j=1}^{K} y_{ij} = 1 \ \forall i$.

**Hàm mất mát:**

$$L(\mathbf{Y}, \mathbf{M}) = \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

**Mục tiêu:** Tìm $\mathbf{Y}, \mathbf{M}$ để tối thiểu hóa $L$.

# Phân tích toán học

**Tối ưu hóa:**

- **Cố định M, tìm Y**:
  - Gán $\mathbf{x}_i$ vào cụm có trung tâm gần nhất:

$$y_{ij} = \begin{cases} 1 & \text{nếu } j = \arg\min_k \|\mathbf{x}_i - \mathbf{m}_k\|_2^2 \\ 0 & \text{ngược lại} \end{cases}$$

- **Cố định Y, tìm M**:
  - Cập nhật trung tâm cụm:

$$\mathbf{m}_j = \frac{\sum_{i=1}^{N} y_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} y_{ij}}$$

  - $\mathbf{m}_j$: Trung bình cộng các điểm trong cụm $j$.

# Các bước trong thuật toán K-means

- Thuật toán lặp lại giữa gán cụm và cập nhật trung tâm cho đến khi hội tụ.
- **Bước 1: Khởi tạo**
  - Chọn số cụm $K$.
  - Khởi tạo ngẫu nhiên $K$ trung tâm $\mathbf{m}_1, \ldots, \mathbf{m}_K$.
  - *Lưu ý*: Sử dụng K-means++ để cải thiện:
    - Chọn ngẫu nhiên một điểm làm trung tâm đầu tiên.
    - Chọn các trung tâm tiếp theo với xác suất tỷ lệ $D(\mathbf{x}_i)^2$.
- **Bước 2: Gán cụm**
  - Với mỗi điểm $\mathbf{x}_i$:
    - Tính khoảng cách đến tất cả trung tâm.
    - Gán vào cụm có trung tâm gần nhất:

$$y_{ij} = \begin{cases} 1 & \text{nếu } j = \arg\min_k \|\mathbf{x}_i - \mathbf{m}_k\|_2^2 \\ 0 & \text{ngược lại} \end{cases}$$

# Các bước trong thuật toán K-means

**Bước 3: Cập nhật trung tâm**

- Với mỗi cụm $j$:
  - Tính lại trung tâm:

$$\mathbf{m}_j = \frac{\sum_{i=1}^{N} y_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} y_{ij}}$$

  - Nếu cụm rỗng, giữ nguyên trung tâm hoặc khởi tạo lại ngẫu nhiên.

**Bước 4: Kiểm tra hội tụ**

- So sánh trung tâm mới với trung tâm cũ.
- Nếu thay đổi $<$ ngưỡng $\varepsilon$ hoặc đạt số vòng lặp tối đa, dừng.
- Ngược lại, quay lại Bước 2.

# Ưu và Nhược điểm của K-means

**Ưu điểm:**

- Đơn giản, dễ triển khai.
- Hiệu quả với dữ liệu lớn, cụm hình cầu.
- Độ phức tạp thấp: $O(N \cdot K \cdot I \cdot d)$, với $I$ là số vòng lặp.
- Phù hợp khi cụm được phân tách rõ ràng.

**Nhược điểm:**

- Phải xác định $K$ trước, khó chọn giá trị tối ưu.
- Nhạy với nhiễu và điểm ngoại lai.
- Không hiệu quả với cụm không hình cầu hoặc kích thước khác nhau.
- Phụ thuộc vào khởi tạo trung tâm, có thể rơi vào cực trị cục bộ.

# DBScan

**DBScan: Density-Based Spatial Clustering of Applications with Noise**

DBScan là thuật toán phân cụm dựa trên mật độ điểm, phổ biến trong học máy và khai phá dữ liệu. Áp dụng tốt trong các trường hợp:

- Phát hiện cụm có hình dạng tự do (không chỉ hình cầu/elip như K-Means, GMM).
- Xử lý tốt dữ liệu nhiễu hoặc ngoại lệ (outliers).

# Các khái niệm quan trọng

**Vùng lân cận epsilon (Eps-neighborhood)** của một điểm dữ liệu $P$ được định nghĩa là tập hợp tất cả các điểm dữ liệu nằm trong phạm vi bán kính epsilon (kí hiệu $\epsilon$) xung quanh điểm $P$.

$$N_{eps}(P) = \{Q \in D \mid d(P,Q) \leq \varepsilon\}$$

**Tiếp cận trực tiếp mật độ (Directly Density-Reachable)** đề cập tới việc một điểm có thể tiếp cận trực tiếp tới một điểm dữ liệu khác. Thỏa mãn 2 điều kiện:

- $Q$ nằm trong vùng lân cận *epsilon* của $P : Q \in N_{eps}(P)$
- Số lượng các điểm dữ liệu nằm trong vùng lân cận *epsilon* tối thiểu là *minPts*: $|N_{eps}(Q)| \geq minPts$

**Tiếp cận mật độ (Density-Reachable):**

- Có chuỗi liên kết từ $P_1 \rightarrow P_n$, mỗi điểm tiếp cận trực tiếp bởi điểm trước đó.

# Các khái niệm quan trọng

**Hai tham số chính của DBScan:**

- $\varepsilon$: Bán kính vùng lân cận.
- minPts: Số điểm tối thiểu trong vùng lân cận để được coi là đủ mật độ.

**Ba loại điểm được xác định:**

- **Điểm lõi (Core Point)**: Có ít nhất minPts điểm trong bán kính $\varepsilon$ (kể cả chính nó).
- **Điểm biên (Border Point)**: Không đủ MinPts trong lân cận bán kính $\varepsilon$, nhưng thuộc vùng lân cận của một điểm lõi.
- **Điểm nhiễu (Noise Point)**: Không thuộc hai loại trên.

# Các bước trong thuật toán DBScanDBScan

Thuật toán lan truyền mở rộng cụm từ điểm lõi đến biên, sau đó chuyển sang cụm mới.

**Bước 1: Khởi tạo**

- Chọn tham số $\varepsilon$ và `minPts`.
- Đánh dấu tất cả điểm là chưa thăm (`visited = False`).
- Đặt chỉ số cụm ban đầu: $C = 0$.

**Bước 2: Duyệt qua từng điểm**

- Nếu điểm chưa được thăm:
  - Đánh dấu đã thăm (`visited = True`).
  - Tìm các láng giềng trong bán kính $\varepsilon$.
  - Nếu số láng giềng $\geq$ `minPts`:
    - Tạo cụm mới: $C = C + 1$
    - Mở rộng cụm từ điểm đó.
  - Nếu số láng giềng $<$ `minPts`, đánh dấu là điểm nhiễu tạm thời.

# Các bước trong thuật toán DBScan

### Bước 3: Mở rộng cụm (Expand Cluster)

- Với mỗi điểm trong lân cận:
  - Nếu chưa được thăm:
    - Đánh dấu đã thăm.
    - Tìm lân cận mới của điểm đó.
    - Nếu số điểm trong lân cận mới $\geq$ `minPts`, thêm vào danh sách mở rộng.
  - Nếu điểm chưa thuộc cụm nào, gán vào cụm hiện tại.

### Bước 4: Lặp lại cho điểm khác

- Khi một cụm được hoàn thành (không còn điểm để xét mở rộng), tiếp tục lặp lại quy trình từ Bước 2 cho các điểm chưa thăm.
- Kết thúc khi tất cả các điểm đều được thăm.

# Ưu và Nhược điểm của DBScan

**Ưu điểm:**

- Không cần chỉ định số cụm trước như K-Means/GMM.
- Phát hiện tốt các cụm có hình dạng bất quy tắc, phức tạp.
- Có khả năng phát hiện điểm nhiễu (noise).

**Nhược điểm:**

- Khó lựa chọn tham số $\varepsilon$ và `minPts` phù hợp.
- Hiệu quả kém nếu dữ liệu có mật độ không đồng đều hoặc các cụm chồng lấn.
- Tính toán khoảng cách cho tất cả cặp điểm $\rightarrow$ chi phí tính toán cao với dữ liệu lớn.

# Mục lục

# K-Nearest Neighbors (KNN)(*)

**KNN là:**

- Thuật toán phân loại và hồi quy đơn giản, mạnh mẽ, dựa trên khoảng cách.
- Thuật toán này thuộc nhóm phương pháp lười học (lazy learning)
- Dựa trên nguyên lý: Các điểm gần nhau có nhiều khả năng thuộc cùng một lớp.

**Ứng dụng thực tiễn đa dạng:**

- Nhận diện chữ viết tay (handwritten character recognition).
- Hệ thống gợi ý (recommendation systems).
- Dự đoán bệnh trong y học (health prediction).
- Phân loại ảnh và video.

# Nguyên lý hoạt động(*)

1. Xác định số lượng điểm $k$ cần xem xét.
2. Tính khoảng cách giữa điểm mới và các điểm trong tập huấn luyện.
3. Chọn $k$ điểm lân cận gần nhất.
4. Phân loại hoặc dự đoán dựa vào thông tin của $k$ điểm.

# Công thức tính khoảng cách(*)

- Khoảng cách Euclidean:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Khoảng cách Manhattan:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

# Các cách chọn nhãn trong KNN(*)

- Có nhiều phương pháp khác nhau để chọn nhãn phù hợp trong bài toán phân loại với KNN. Dưới đây là các phương pháp phổ biến:

# 1. Đa số phiếu (Majority Voting)(*)

- Chọn nhãn xuất hiện nhiều nhất trong $k$ điểm lân cận.
- Công thức:

$$\text{Label} = \arg\max_{c} \sum_{i=1}^{k} \mathbb{I}(y_i = c)$$

- **Ưu điểm**: Đơn giản, dễ hiểu.
- **Nhược điểm**: Có thể bị ảnh hưởng bởi các lớp chiếm ưu thế (lớp có số điểm nhiều hơn).

# 2. Trọng số theo khoảng cách (Distance-Weighted Votin(*)

- Gán trọng số dựa trên khoảng cách: Điểm gần hơn có trọng số cao hơn.
- Công thức:

$$\text{Label} = \arg\max_{c} \sum_{i=1}^{k} \frac{\mathbb{I}(y_i = c)}{d(x, x_i)}$$

- **Ưu điểm**: Giảm ảnh hưởng của điểm xa.
- **Nhược điểm**: Cần tính toán khoảng cách chính xác, có thể tốn thời gian.

# Giới thiệu Naive Bayes Classifier(*)

Naive Bayes Classifier là:

- Thuật toán phân loại đơn giản, mạnh mẽ, dựa trên **Định lý Bayes**.
- Giả định các đặc trưng **độc lập** khi biết lớp dữ liệu.

Ứng dụng thực tiễn đa dạng:

- Phân loại thư rác (email spam).
- Phân tích cảm xúc (bình luận, đánh giá).
- Phân loại văn bản, dữ liệu chiều cao.

# Xây dựng Naive Bayes Classifier: Bài toán phân loại(*)

Trong bài toán phân loại chúng ta có $C$ lớp (class): $1, 2, \ldots, C$. Với mỗi điểm dữ liệu $\mathbf{x} \in \mathbb{R}^d$ (vector đầu vào $d$ chiều).

Mục tiêu: Xác định xác suất $\mathbf{x}$ thuộc lớp $k \in \{1, 2, \ldots, C\}$:

$$p(y = k|\mathbf{x}) = p(k|\mathbf{x}).$$

Sau khi tính được xác xuất p(k|x) cho từng lớp, ta gán điểm dữ liệu $x$ vào lớp có xác suất cao nhất, tức là:

$$y = c^* = \arg \max_{k \in \{1,\ldots,C\}} p(k \mid \mathbf{x}).$$

# Xây dựng Naive Bayes Classifier: Quy tắc Bayes(*)

Biểu thức $p(k \mid \mathbf{x})$ thường khó tính trực tiếp. Vì vậy ta áp dụng quy tắc Bayes để chuyển biểu thức về dạng dễ tính hơn:

$$p(k|\mathbf{x}) = \frac{p(\mathbf{x}|k)p(k)}{p(\mathbf{x})}.$$

Do $p(\mathbf{x})$ không phụ thuộc vào $k$, ta có thể bỏ qua và chỉ cần tối ưu hóa phần tử ở tử số:

$$y = c^* = \arg \max_{k \in \{1,\dots,C\}} p(\mathbf{x}|k)p(k).$$

Trong đó:

- $p(k)$: Xác suất để một mẫu dữ liệu bất kỳ rơi vào lớp $k$. Thường được ước lượng bằng *Maximum Likelihood Estimation* (MLE), tức là tỷ lệ số lượng điểm dữ liệu trong lớp k so với tổng số điểm trong tập huấn luyện.
- $p(\mathbf{x} \mid k)$: Xác suất để tồn tại điểm dữ liệu $\mathbf{x}$ với điều kiện nó thuộc lớp $k$. Đây là một dữ liệu khó tính toán vì $\mathbf{x}$ có thể có nhiều chiều (nhiều đặc trưng).

# Xây dựng Naive Bayes Classifier: Giả định độc lập(*)

**Giả thiết Naive (Naive's assumption)** Các thành phần của $\mathbf{x}$ (*features*) độc lập với nhau trong mọi phân lớp $k$.

Biểu thức xác suất điều kiện:

$$p(\mathbf{x}|k) = p(x_1, x_2, \ldots, x_d|k) = \prod_{i=1}^{d} p(x_i|k).$$

Giả định đơn giản, không luôn đúng nhưng hiệu quả trong nhiều ứng dụng.

Giảm đáng kể độ phức tạp tính toán.

# Naive Bayes Classifier: Quá trình kiểm tra(*)

Với điểm dữ liệu mới $\mathbf{x}$, xác định lớp:

$$y = c^* = \arg \max_{k \in \{1,\dots,C\}} p(k) \prod_{i=1}^{d} p(x_i|k).$$

Để tránh sai số với xác suất nhỏ, dùng logarit:

$$y = c^* = \arg \max_{k \in \{1,\dots,C\}} \left[ \log p(k) + \sum_{i=1}^{d} \log p(x_i|k) \right].$$

Log là hàm đồng biến nên không ảnh hưởng kết quả.

# Naive Bayes Classifier(*)

**Tính** $p(x_i \mid k)$ - $i = 1, \ldots, d; k = 1, 2 \ldots, C$

Cách tính $p(x_i \mid k)$ phụ thuộc vào kiểu dữ liệu $x_i$.

- $x_i$ - Category: Dữ liệu có kiểu phân loại (lựa chọn)
  - $x_i$ - Multinomial (Nhiều lựa chọn): Phương pháp *Multinomial Naive Bayes*.
  - $x_i$ - Binary (2 lựa chọn): Phương pháp *Bernoulli Naive Bayes*.
- $x_i$ - Numeric: Phương pháp *Gaussian Naive Bayes*

# Phân phối: Gaussian Naive Bayes(*)

Áp dụng cho dữ liệu liên tục, với giả định mỗi đặc trưng $x_i$ trong lớp k có phân phối chuẩn.

Xác suất có điều kiện:

$$p(x_i|k) = \frac{1}{\sqrt{2\pi\sigma_{k,i}^2}} \exp\left(-\frac{(x_i - \mu_{k,i})^2}{2\sigma_{k,i}^2}\right),$$

trong đó:

- $\mu_{k,i}$: Trung bình của đặc trưng $x_i$ trong lớp $k$.
- $\sigma_{k,i}^2$: Độ lệch chuẩn của đặc trưng $x_i$ trong lớp $k$.

# Phân phối: Multinomial Naive Bayes(*)

Áp dụng cho dữ liệu rời rạc, với các đặc trưng $x_i$ có kiểu nhiều lựa chọn.

VD: nhóm máu, ngày trong tuần, tần suất từ,...

Xác suất có điều kiện:

$$p(x_i|k) = \frac{N_{k,i} + \alpha}{N_k + \alpha d},$$

trong đó:

- $N_{k,i}$: Số lần đặc trưng $x_i$ xuất hiện trong lớp $k$.
- $N_k$: Tổng số lần xuất hiện của tất cả đặc trưng trong lớp $k$.
- $d$: Số lượng đặc trưng.
- $\alpha$: Tham số điều chỉnh (thường $\alpha = 1$ cho Laplace smoothing).

# Phân phối: Bernoulli Naive Bayes(*)

Áp dụng cho dữ liệu dạng 02 lựa chọn.

VD: Giới tính; Có/Không hút thuốc....

Xác suất có điều kiện:

$$p(x_i|k) = p(x_i = 1|k)^{x_i} \cdot (1 - p(x_i = 1|k))^{1-x_i},$$

trong đó:

- $p(x_i = 1|k)$: Xác suất đặc trưng $x_i = 1$ trong lớp $k$.
- $x_i$: Giá trị đặc trưng (0 hoặc 1).

# Ứng dụng: Nhận diện American Sign Language (ASL)(

**Dữ liệu đầu vào**:

- Hình ảnh 2D ký hiệu ASL (A-Y, trừ J, Z), sau đó mỗi hình ảnh được "duỗi" thành một vector 1 chiều, với mỗi phần tử trong vector đại diện cho giá trị cường độ pixel tại một vị trí cụ thể
- Giá trị pixel grayscale ([0,1]), liên tục, không giới hạn ở mức rời rạc.

**Tại sao chọn Gaussian Naive Bayes?**:

- Phù hợp với dữ liệu liên tục, giả định pixel tuân theo phân phối chuẩn.
- Tính xác suất có điều kiện của pixel dựa trên trung bình và phương sai, hiệu quả cho phân loại ASL.
- Hoạt động tốt dù giá trị pixel không hoàn toàn chuẩn, nhờ tính đơn giản.

# Softmax Regression là gì?(*)

- Là một thuật toán phân loại tuyến tính (linear classification algorithm).
- Là sự mở rộng của Logistic Regression cho bài toán có **nhiều hơn hai lớp** (multi-class classification).
- Thích hợp cho các bài toán mà **mỗi mẫu chỉ thuộc về DUY NHẤT một lớp**.
- Đầu ra là phân phối xác suất cho từng lớp, cho biết khả năng mẫu thuộc về lớp nào.

# Nguyên lý Hoạt động: Tính điểm (Scores)(*)

- Mỗi hình ảnh đầu vào (sau khi xử lý) được biểu diễn dưới dạng một vector đặc trưng $x$.
- Mô hình học một bộ trọng số $W$ (ma trận) và độ lệch $b$ (vector).
- Đối với mỗi lớp $k$ (trong bài toán ASL là 24 lớp A-Z trừ J, Z), mô hình tính một điểm (score) hoặc logit $z_k$ bằng cách kết hợp tuyến tính:

$$z_k = w_k^T x + b_k$$

- Dạng ma trận:

$$Z = Wx + b$$

  Trong đó:
  - $x \in \mathbb{R}^d$ là vector đặc trưng đầu vào (ví dụ: pixel của ảnh).
  - $W \in \mathbb{R}^{K \times d}$ là ma trận trọng số.
  - $b \in \mathbb{R}^K$ là vector độ lệch.
  - $Z \in \mathbb{R}^K$ là vector điểm cho từng lớp.
- Các điểm này có thể là bất kỳ giá trị thực nào (dương, âm hoặc không).

# One-hot Coding(*)

- Với cách biểu diễn one-hot, mỗi output không còn là một giá trị tương ứng với mỗi class nữa mà là một **vector** có đúng một phần tử bằng 1, các phần tử còn lại bằng 0.
- Phần tử bằng 1 nằm ở vị trí tương ứng với class đó, thể hiện rằng điểm dữ liệu đang xét rơi vào class này với xác suất bằng 1 (vì đây là sự thật, không cần dự đoán).

# Nguyên lý Hoạt động: Hàm Softmax(*)

- Để biến các điểm $z_k$ thành xác suất, chúng ta sử dụng hàm Softmax.
- Hàm Softmax chuẩn hóa các điểm sao cho chúng nằm trong khoảng $(0, 1)$ và tổng bằng 1.
- Công thức:

$$P(y = k \mid x; W, b) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

  Trong đó:
  - $e^{z_k}$ là lũy thừa của điểm $z_k$ (luôn dương).
  - $\sum_{j=1}^{K} e^{z_j}$ là tổng chuẩn hóa.
- Vector xác suất: $[\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_K]$, với $\sum \hat{y}_k = 1$.
- Dự đoán là lớp có xác suất cao nhất:

$$\hat{y} = \arg\max_k P(y = k \mid x; W, b)$$

# Nguyên lý Hoạt động: Hàm mất mát và Huấn luyện(*)

- Hàm mất mát dùng để đo sự khác biệt giữa dự đoán và nhãn thực tế.
- Dùng hàm **Cross-Entropy Loss**:

$$L(\hat{y}, y) = -\sum_{k=1}^{K} y_k \log(\hat{y}_k)$$

- Mục tiêu: giảm trung bình loss trên toàn bộ tập dữ liệu:

$$J(W, b) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log(\hat{y}_{ik})$$

- Dùng Gradient Descent để cập nhật:

$$W \leftarrow W - \alpha \nabla_W L, \quad b \leftarrow b - \alpha \nabla_b L$$

# Công thức Tổng quát(*)

- **Đầu vào:** $x \in \mathbb{R}^d$
- **Trọng số và độ lệch:** $W \in \mathbb{R}^{K \times d}$, $b \in \mathbb{R}^K$
- **Logits:** $Z = Wx + b$
- **Softmax:**

$$P(y = k \mid x; W, b) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

- **Cross-Entropy Loss:**

$$L(W, b; x, y) = -\sum_{k=1}^{K} y_k \log \left( \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \right)$$

- **Cập nhật:**

$$W \leftarrow W - \alpha \nabla_W L, \quad b \leftarrow b - \alpha \nabla_b L$$

# Áp dụng Softmax Regression cho Nhận diện ASL(*)

- **Dữ liệu:** hình ảnh bàn tay biểu diễn chữ cái ASL.
- **Số lớp:** $K = 24$
- **Mô hình:**
  - Đầu vào: $d = 1024$, đầu ra: $K = 24$
  - Kích hoạt Softmax ở đầu ra.
- **Huấn luyện:** sử dụng Cross-Entropy và Gradient Descent
- **Dự đoán:** lấy lớp có xác suất cao nhất

# Mục lục

# Giới thiệu

- Bài toán: phân loại ảnh ngôn ngữ ký hiệu ASL.
- Thông thường là bài toán phân loại nhãn rời rạc (chữ cái A-Z).
- Tuy nhiên, ta biến đổi bài toán phân loại sang bài toán hồi quy:
  - Chọn ngẫu nhiên một lớp.
  - Dùng xác suất softmax của lớp đó làm nhãn hồi quy.
- Mô hình thử nghiệm: **Linear Regression** và **Decision Tree**.

# Ý tưởng mô hình hồi quy

- Thay vì phân loại nhãn rời rạc, ta chọn ngẫu nhiên một lớp và dùng xác suất softmax làm nhãn hồi quy.
- Mục tiêu: Hồi quy dự đoán xác suất một ảnh thuộc lớp đó.
- Áp dụng hai mô hình: Hồi quy tuyến tính và Cây quyết định.

# Kết quả thực nghiệm (Tóm tắt)

- So sánh Linear Regression và Decision Tree với và không dùng PCA.
- Ba tỷ lệ train:test: 8:2, 7:3, 6:4.
- Đánh giá qua: Accuracy, Precision, Recall.

# Bảng kết quả

| Mô hình | PCA | Tỉ lệ | Accuracy | Precision | Recall |
|---|---|---|---|---|---|
| 6*Linear Regression | Không | 8:2 | 0.988 | 0.989 | 0.759 |
| | | 7:3 | 0.987 | 0.978 | 0.735 |
| | | 6:4 | 0.988 | 0.967 | 0.775 |
| | Có | 8:2 | 0.993 | 0.996 | 0.843 |
| | | 7:3 | 0.953 | 0.996 | 0.842 |
| | | 6:4 | 0.592 | 0.955 | 0.841 |
| 6*Decision Tree | Không | 8:2 | 0.982 | 0.830 | 0.767 |
| | | 7:3 | 0.984 | 0.867 | 0.765 |
| | | 6:4 | 0.981 | 0.830 | 0.754 |
| | Có | 8:2 | 0.991 | 0.890 | 0.908 |
| | | 7:3 | 0.990 | 0.893 | 0.893 |
| | | 6:4 | 0.990 | 0.890 | 0.898 |

# Phân tích kết quả

**So sánh giữa Linear Regression và Decision Tree**

- Decision Tree thường cho Accuracy và Recall cao hơn, nhất là khi có PCA.
- Linear Regression gặp khó với dữ liệu phi tuyến tính.
- Decision Tree khai thác tốt đặc trưng phi tuyến từ ảnh.

# Phân tích kết quả

**Ảnh hưởng của PCA**

- Linear Regression: PCA giúp tăng Recall, nhưng đôi khi làm giảm Accuracy (ví dụ 6:4).
- Decision Tree: PCA giúp cải thiện toàn diện (Accuracy, Precision, Recall).
- PCA giúp giảm chiều, chống overfitting, làm mô hình ổn định hơn.

# Phân tích kết quả

**Ảnh hưởng của tỷ lệ train:test**

- Accuracy giảm nhẹ khi giảm dữ liệu huấn luyện.
- Recall tăng khi dữ liệu kiểm tra đa dạng hơn.
- Tác động tiêu cực có thể được giảm bằng PCA và mô hình phù hợp.

# Kết luận

- **Decision Tree + PCA** là phương án hiệu quả nhất.
- PCA hữu ích với Decision Tree, nhưng có thể gây nhiễu cho Linear Regression.
- Cần chọn mô hình và tiền xử lý phù hợp với bản chất dữ liệu.

# Mục lục

# Phân cụm: KMeans

Thuật toán KMeans được áp dụng trên dữ liệu gốc (sau chuẩn hóa, chưa PCA) với $k = 24$ (bằng số nhãn thực tế).

**Chỉ số đánh giá:**

- **Silhouette Score:** 0.1654
- **Adjusted Rand Index (ARI):** 0.0426
- **Normalized Mutual Information (NMI):** 0.2048

# Nhận xét về Chỉ Số KMeans

**Nhận xét:**

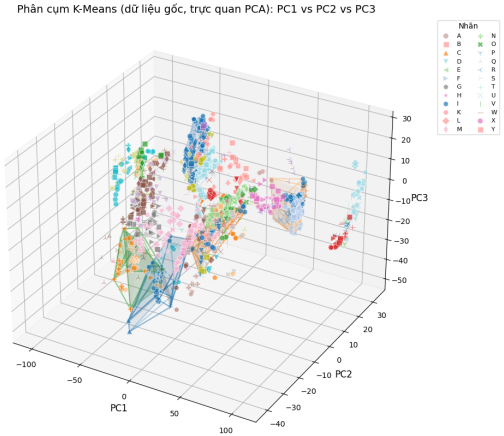
- **Silhouette Score = 0.1654:** Thấp, các cụm chồng lấn, ranh giới không rõ ràng.
- **ARI = 0.0426:** Gần 0, tương đồng thấp với nhãn thực tế, gán nhãn gần ngẫu nhiên.
- **NMI = 0.2048:** Thấp, thông tin chung hạn chế, cụm không đại diện tốt lớp thực tế.

**Kết luận:** KMeans không phù hợp với dữ liệu ký hiệu tay trên không gian đặc trưng gốc do cấu trúc phức tạp và chồng lấn lớp.

# Trực Quan Hóa KMeans bằng PCA 3D

**Nhận xét:**

- Cụm chồng lấn, không tách biệt rõ theo nhãn.
- Nhiều cụm phân tán rộng.
- Phù hợp với chỉ số thấp (Silhouette, ARI, NMI).



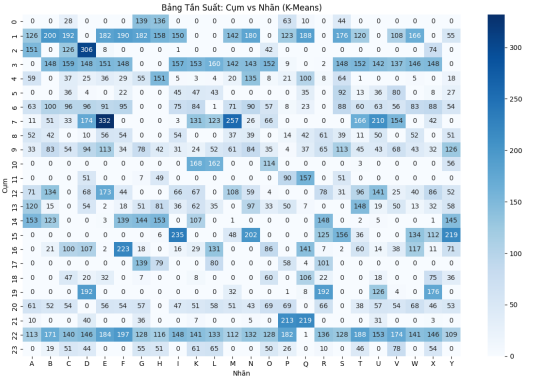

Hình: Phân cụm KMeans trên dữ liệu gốc, trực

# Bảng Tần Số Cụm KMeans và Nhãn Gốc

**Nhận xét:**

- Nhiều cụm chứa nhiều nhãn, pha trộn mạnh (ví dụ: cụm 0, 1, 2...).
- Không có đường chéo nổi bật, xác nhận sự pha trộn nhãn trong cụm.

**Kết luận:** KMeans không tách biệt hiệu quả lớp ký hiệu tay do cụm pha trộn và rời rạc.




Hình: Bảng tần suất giữa cụm KMeans và nhãn gốc. Hàng: cụm KMeans, cột: nhãn gốc, ô: số điểm dữ liệu.

# Phân cụm: DBScan

Thuật toán DBSCAN được áp dụng với tham số:

- **Eps:** 38.44
- **MinPts:** 45

**Kết quả:**

- **Số cụm:** 55
- **Tỷ lệ nhiễu:** 8.56%
- **Silhouette Score:** 0.1714
- **Adjusted Rand Index (ARI):** 0.0157
- **Normalized Mutual Information (NMI):** 0.2121

# Nhận xét về Kết Quả DBSCAN

**Nhận xét:**

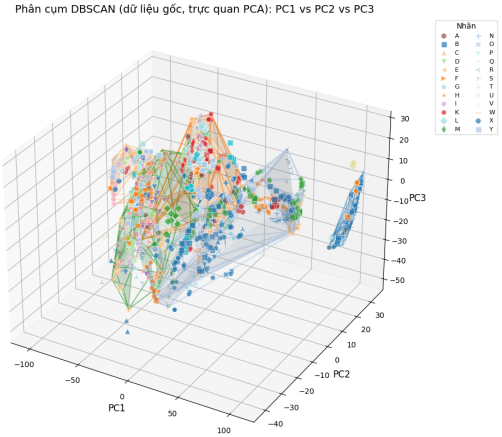
- 55 cụm nhỏ, phân mảnh quá mức (over-segmentation).
- **Tỷ lệ nhiễu (8.56%):** Hợp lý, một phần nhỏ dữ liệu bị loại.
- **Silhouette Score = 0.1714:** Thấp, cụm yếu, chồng lấn.
- **ARI = 0.0157:** Gần 0, tương quan thấp với nhãn gốc.
- **NMI = 0.2121:** Thấp, thông tin chung hạn chế.

**Kết luận:** DBSCAN không phù hợp do dữ liệu phức tạp, mật độ không đồng đều, và lớp chồng lấn.

# Trực Quan Hóa DBSCAN bằng PCA 3D

**Nhận xét:**

- Một số cụm (như bên phải) chặt chẽ, ít chồng lấn.
- Các cụm khác phân tán, trùng lặp nhãn gốc.
- DBSCAN phát hiện cấu trúc tiềm ẩn mà không cần $k$.



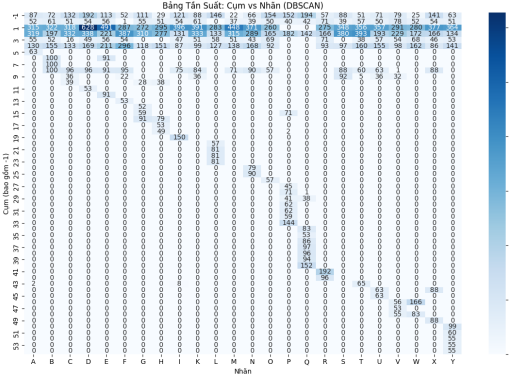

Hình: Phân cụm DBSCAN trên dữ liệu gốc

# Bảng Tần Số Cụm DBSCAN và Nhãn Gốc

**Nhận xét:**

- Cụm lớn (0, 1, 2, 3) chứa nhiều nhãn, chồng lấn.
- Cụm nhỏ (6, 7, 8, 9) thuần nhất, tách biệt tốt.
- Cụm -1 (noise) chứa nhiều nhãn khác nhau, có thể là outlier.

**Kết luận:** DBSCAN hiệu quả với cụm mật độ rõ, nhưng yếu với lớp chồng lấn.




Hình: Bảng tần suất giữa cụm DBSCAN và nhãn gốc. Hàng: cụm DBSCAN, cột: nhãn gốc, ô: số điểm dữ liệu.

# Bảng tổng hợp và phân tích kết quả

**Tổng hợp chỉ số:**

| Mô hình | Train Time (s) | Accuracy | Precision | Recall |
|---|---|---|---|---|
| GNB | 0.0316 | 0.5939 | 0.6616 | 0.5939 |
| MLR | 1.4367 | 0.7680 | 0.7706 | 0.7680 |
| KNN | 0.0147 | 0.9928 | 0.9928 | 0.9928 |

**Phân tích:**

- **Train time:** KNN nhanh nhất (0.0147s), GNB thứ hai (0.0316s), MLR chậm nhất (1.4367s).
- **Accuracy:** KNN vượt trội (99.28%), MLR khá tốt (76.80%), GNB thấp nhất (59.39%).
- **Precision, Recall:** KNN cao và cân bằng, MLR ổn, GNB kém do giả định độc lập mạnh.