ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - CƠ - TIN HỌC

Học máy

Giảng viên: Cao Văn Chung

Mã lớp học phần: MAT1201

Nhận diện bảng chữ cái ngôn ngữ ký hiệu

Thành viên nhóm

Nguyễn Vĩ Chí Cường	(22000073)
Đỗ Tiến Dũng	(22000075)
Đỗ Hữu Đạt	(22000080)

Hà Nội, 2025

Lời nói đầu

Trong thời đại công nghệ số phát triển mạnh mẽ, trí tuệ nhân tạo (AI) và học máy (Machine Learning) ngày càng đóng vai trò quan trọng trong nhiều lĩnh vực của đời sống, đặc biệt là trong việc hỗ trợ giao tiếp và nâng cao chất lượng cuộc sống cho cộng đồng. Một trong những ứng dụng đầy tiềm năng của học máy là khả năng nhận diện ngôn ngữ ký hiệu – một phương tiện giao tiếp thiết yếu đối với người khiếm thính.

Đề tài này tập trung vào bài toán nhận diện ký hiệu tay trong ngôn ngữ ký hiệu Mỹ (ASL - American Sign Language) thông qua hình ảnh. Với sự hỗ trợ của bộ dữ liệu "ASL Handsign Dataset (Grayscaled Thresholded)" từ nền tảng Kaggle, các phương pháp học máy và học sâu sẽ được áp dụng để xây dựng mô hình phân loại ký hiệu tay một cách hiệu quả và chính xác.

Thông qua đề tài này, người thực hiện mong muốn không chỉ làm quen và ứng dụng các kỹ thuật trong lĩnh vực thị giác máy tính mà còn góp phần nhỏ vào việc nghiên cứu các giải pháp công nghệ giúp hỗ trợ người khiếm thính trong việc giao tiếp và hòa nhập với cộng đồng.

Nội dung đề tài được trình bày thành các chương bao gồm: giới thiệu bộ dữ liệu, tiền xử lý, trực quan hóa, xây dựng mô hình phân loại, đánh giá hiệu suất mô hình và kết luận. Qua đó, đề tài nhằm mục tiêu khái quát toàn bộ quy trình xây dựng một hệ thống nhận diện ký hiệu tay dựa trên hình ảnh, từ bước thu thập dữ liệu đến triển khai mô hình.

Mục lục

1 Giới thiệu bộ dữ liệu

1.1 Tổng quan về bộ dữ liệu ASL Handsign

Tập dữ liệu ASL Handsign Dataset (Grayscaled & Thresholded) là một tập dữ liệu hình ảnh được thiết kế cho bài toán nhận dạng ngôn ngữ ký hiệu Mỹ (ASL - American Sign Language). Mỗi hình ảnh trong tập dữ liệu thể hiện một ký hiệu tay tương ứng với các chữ cái trong bảng chữ cái tiếng Anh. Bộ dữ liệu đã được xử lý dưới dạng ảnh Grayscaled và nhị phân (thresholded), giúp tăng hiệu quả khi huấn luyện các mô hình học máy và học sâu.

Tập dữ liệu này rất phù hợp cho các bài toán phân loại ảnh trong lĩnh vực thị giác máy tính, đặc biệt là các bài toán về nhận diện ký hiệu ngôn ngữ.

1.2 Nguồn dữ liệu

Bộ dữ liệu được tải từ trang Kaggle với đường dẫn:

https://www.kaggle.com/datasets/furkanakdeniz/asl-handsign-dataset-grayscaled-thresholded

Bộ dữ liệu được chia sẻ công khai bởi người dùng furkanakdeniz.

1.3 Cấu trúc của bộ dữ liệu

Bộ dữ liệu được tổ chức thành hai thư mục chính:

- train/: Bao gồm các hình ảnh dùng để huấn luyện mô hình. Mỗi ảnh tương ứng với một chữ cái từ A đến Y (trừ J và Z – là các ký hiệu động, không được thể hiện bằng ảnh tĩnh). Mỗi lớp (class) là một thư mục chứa nhiều ảnh đại diện cho ký hiệu tương ứng. Số lượng ảnh train: Grayscale- 22.880 | Threshold-30.050
- test/: Chứa tập ảnh kiểm tra, được tổ chức tương tự như tập huấn luyện. Số lượng ảnh test Grayscale- 4.053 | Threshold-7.523
- Đặc điểm của các hình ảnh trong bộ dữ liệu:
  - Ảnh đen trắng (grayscale) đã được xử lý threshold để làm nổi bật phần ký hiệu tay.
  - Kích thước ảnh: 128x128 pixel, thuận tiện khi áp dụng các kiến trúc CNN đơn giản.
  - Mỗi ảnh thuộc về một trong 24 lớp (từ A đến Y, ngoại trừ J và Z).

2 Phương pháp tiền xử lý dữ liệu

2.1 Resize ảnh (Sử dụng hàm resize của thư viện opencv)

Resize ảnh là một bước quan trọng trong tiền xử lý dữ liệu trước khi huấn luyện mô hình học máy, đặc biệt trong bài toán nhận diện bảng chữ cái ngôn ngữ ký hiệu American Sign Language (ASL) với cách tiếp cận chuyển hình ảnh 2D thành vector các pixel.

2.1.1 Mục đích

Biến đổi hình ảnh từ các kích thước ban đầu thành một kích thước chuẩn nhất định, giúp tạo ra sự đồng nhất về số chiều đặc trưng và cải thiện hiệu năng của mô hình học máy.

2.1.2 Tác động

- **Chuẩn hóa kích thước ảnh**: Đảm bảo tất cả hình ảnh có cùng kích thước (ví dụ: 28×28), tạo vector đặc trưng có độ dài cố định (784 chiều), phù hợp với đầu vào của Gaussian Naive Bayes.
- **Giảm số chiều đặc trưng**: Giảm số lượng pixel (ví dụ: từ $64 \times 64 = 4096$ xuống $28 \times 28 = 784$), giảm chi phí tính toán, nhiễu, và cải thiện khả năng tổng quát hóa của mô hình.
- **Bảo toàn thông tin hình ảnh**: Nội suy tuyến tính kép giữ chi tiết hình dạng tay, đảm bảo các ký hiệu ASL (như "A" và "E") vẫn phân biệt được, hỗ trợ Gaussian Naive Bayes mô hình hóa dữ liệu hiệu quả hơn.
- **Cải thiện hiệu suất huấn luyện**: Kích thước ảnh đồng nhất và số chiều giảm giúp quá trình tính toán xác suất trong Gaussian Naive Bayes nhanh hơn, đồng thời giảm nguy cơ sai số số học khi số chiều lớn.

2.2 Chuẩn hóa dữ liệu

Chuẩn hóa dữ liệu là một bước quan trọng trong tiền xử lý dữ liệu trước khi huấn luyện mô hình học máy, đặc biệt trong bài toán nhận diện bảng chữ cái ngôn ngữ ký hiệu American Sign Language (ASL) với cách tiếp cận chuyển hình ảnh 2D thành vector các pixel.

2.2.1 Mục đích

Biến đổi dữ liệu từ các phạm vi giá trị ban đầu thành các phạm vi giá trị theo một chuẩn nhất định, giúp tạo ra sự cân bằng giữa các đặc trưng và cải thiện hiệu năng của mô hình học máy.

2.2.2 Phương pháp Standardization (Z-score normalization)

Nội dung phương pháp: Phương pháp này sẽ chuyển đổi dữ liệu về một phân bố trong đó giá trị trung bình bằng 0 và độ lệch chuẩn bằng 1.

Công thức chuẩn hóa:

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

trong đó:

– $\text{mean}(X)$ là kỳ vọng (giá trị trung bình) của X,

– $\text{std}(X)$ là độ lệch chuẩn của X.

Ví dụ: Với tập dữ liệu X có $\text{mean}(X) = 10$, $\text{std}(X) = 4$, giá trị $x = 15.9$ sẽ được chuẩn hóa thành:

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

2.2.3 Tác động

- **Hỗ trợ quá trình huấn luyện học máy hội tụ nhanh hơn**: Chuẩn hóa dữ liệu làm cho các đặc trưng có cùng phạm vi giá trị, từ đó gradient của các đặc trưng sẽ có tầm ảnh hưởng tương đồng, giúp cho quá trình cập nhật tham số diễn ra một cách đồng đều và tránh được tình trạng *overshooting*, tạo điều kiện thuận lợi cho quá trình hội tụ tới lời giải tối ưu.
- **Đảm bảo cân bằng đặc trưng**: Các đặc trưng có phạm vi giá trị lớn hơn sẽ ảnh hưởng lớn hơn đến hàm mất mát, khiến mô hình thiên về các đặc trưng đó tạo ra một mô hình không cân bằng. Khi dữ liệu được chuẩn hóa, mọi đặc trưng đều có cùng phạm vi giá trị và cùng mức biến động, giúp khắc phục tình trạng trên.
- **Dễ dàng hơn trong điều chỉnh tham số**: Chuẩn hóa dữ liệu giúp làm giảm sự biến động của gradient, từ đó việc tinh chỉnh tốc độ học cho phù hợp với độ lớn của gradient trở nên dễ dàng hơn, giúp quá trình huấn luyện tránh xảy ra bất ổn.
- **Cải thiện độ chính xác cho mô hình**: Khi dữ liệu được chuẩn hóa, các mối quan hệ phức tạp trong dữ liệu được mô hình hóa hiệu quả hơn, từ đó cải thiện độ chính xác của mô hình.

2.3 Phân tích thành phần chính (PCA)

Phân tích thành phần chính (PCA) là một bước quan trọng trong tiền xử lý dữ liệu trước khi huấn luyện mô hình học máy, đặc biệt trong bài toán nhận diện bảng chữ cái ngôn ngữ ký hiệu American Sign Language (ASL) với cách tiếp cận chuyển hình ảnh 2D thành vector các pixel.

2.3.1 Mục đích

Giảm số chiều của dữ liệu bằng cách biến đổi các đặc trưng ban đầu thành một tập hợp các thành phần chính mới, giúp loại bỏ nhiễu, giảm chi phí tính toán và cải thiện hiệu năng của mô hình học máy.

2.3.2 Phương pháp PCA

Nội dung phương pháp: Phương pháp này tìm ra các thành phần chính (principal components) là các hướng trong không gian dữ liệu có phương sai lớn nhất, sau đó chiếu

dữ liệu gốc lên các hướng này để giảm số chiều.

Các bước thực hiện:

- Chuẩn hóa dữ liệu: Đưa dữ liệu về phân bố có trung bình bằng 0 và độ lệch chuẩn bằng 1.
- Tính ma trận hiệp phương sai: Ma trận này thể hiện mức độ tương quan giữa các đặc trưng.
- Phân tích giá trị riêng và vector riêng: Tìm các giá trị riêng và vector riêng của ma trận hiệp phương sai, trong đó các vector riêng tương ứng với các thành phần chính.
- Sắp xếp các thành phần chính: Sắp xếp các vector riêng theo thứ tự giảm dần của giá trị riêng, chọn k thành phần chính đầu tiên (với k nhỏ hơn số chiều ban đầu).
- Chiếu dữ liệu: Chiếu dữ liệu gốc lên không gian mới được tạo bởi k thành phần chính.

Công thức chiếu dữ liệu: Giả sử dữ liệu gốc là X (ma trận $n \times d$ với n mẫu và d chiều), ma trận thành phần chính là W (ma trận $d \times k$ chứa k vector riêng), dữ liệu sau khi giảm chiều là:

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

trong đó:

- X_{PCA} là ma trận $n \times k$ chứa dữ liệu sau khi giảm chiều,
- W là ma trận chứa k thành phần chính (các vector riêng).

Ví dụ: Với vector pixel ban đầu có kích thước 784 chiều (từ ảnh 28×28), áp dụng PCA để giảm xuống 50 chiều. Giả sử sau khi tính toán, $k = 50$ thành phần chính được chọn, giữ lại 95% phương sai của dữ liệu. Dữ liệu sau khi chiếu sẽ có kích thước $n \times 50$, với n là số mẫu.

2.3.3 Tác động

- **Giảm số chiều đặc trưng**: Trong bài toán ASL, PCA giảm số chiều từ 784 (ảnh 28×28) xuống còn 50, giảm chi phí tính toán và nhiễu, giúp mô hình Gaussian Naive Bayes hoạt động hiệu quả hơn.
- **Loại bỏ nhiễu và giữ thông tin chính**: PCA tập trung vào các thành phần có phương sai lớn nhất, loại bỏ các đặc trưng ít quan trọng (nhiễu), giữ lại thông tin chính về hình dạng tay trong các ký hiệu ASL.
- **Cải thiện hiệu suất huấn luyện**: Số chiều giảm làm giảm số lượng tham số cần ước lượng trong Gaussian Naive Bayes (như $\mu_{c,i}$, $\sigma^2_{c,i}$), giúp quá trình huấn luyện nhanh hơn và giảm nguy cơ quá khớp.
- **Hỗ trợ trực quan hóa dữ liệu**: Khi giảm xuống 2 hoặc 3 chiều, PCA cho phép trực quan hóa dữ liệu, giúp phân tích sự phân tách giữa các lớp (các ký hiệu từ A đến Z) trong không gian mới.

3 Mô hình KNN

3.1 Giới thiệu mô hình

Thuật toán K-Nearest Neighbors (KNN) là một thuật toán học có giám sát dùng để dự đoán nhãn của một điểm dữ liệu mới dựa trên nhãn của k điểm láng giềng gần nhất trong tập huấn luyện. Thuật toán này thuộc nhóm phương pháp học lười (*lazy learning*), tức là không xây dựng mô hình trong giai đoạn huấn luyện mà chỉ thực hiện tính toán khi cần dự đoán. KNN được đánh giá cao vì tính dễ hiểu, dễ triển khai và không yêu cầu giả định về phân phối dữ liệu. Nó phụ thuộc vào khoảng cách giữa các điểm dữ liệu và thường sử dụng các thước đo như khoảng cách Euclid, Manhattan hoặc Minkowski.

3.2 Xây dựng mô hình

3.2.1 Tập dữ liệu

Tập huấn luyện bao gồm n điểm dữ liệu có nhãn, được biểu diễn dưới dạng:

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

Trong đó, mỗi $x_i \in R^p$ là một vector đặc trưng p chiều, và $y_i \in \{C_1, C_2, \ldots, C_m\}$ là nhãn lớp, với m là số lớp. Điểm dữ liệu mới cần phân loại là x.

3.2.2 Tính khoảng cách

Khoảng cách giữa điểm mới x và các điểm x_i trong tập huấn luyện được tính bằng các công thức phổ biến:

- **Khoảng cách Euclid**:

$$d(x, x_i) = \sqrt{\sum_{j=1}^{p} (x_j - x_{ij})^2}$$

- **Khoảng cách Manhattan**:

$$d(x, x_i) = \sum_{j=1}^{p} |x_j - x_{ij}|$$

- **Khoảng cách Minkowski**:

$$d(x, x_i) = \left(\sum_{j=1}^{p} |x_j - x_{ij}|^q \right)^{1/q}$$

  với $q \geq 1$, trong đó $q = 2$ tương ứng với Euclid và $q = 1$ tương ứng với Manhattan.

3.2.3 Chuẩn hóa dữ liệu

Chuẩn hóa dữ liệu là bước cần thiết để đảm bảo các đặc trưng có đóng góp công bằng khi tính khoảng cách, tránh trường hợp đặc trưng có giá trị lớn chi phối kết quả.

3.2.4 Giá trị k

- Giá trị k ảnh hưởng lớn đến hiệu suất mô hình:
  - k nhỏ: Mô hình nhạy cảm với nhiễu, dễ bị *overfitting*.
  - k lớn: Mô hình tổng quát hơn nhưng có thể bỏ sót chi tiết, dẫn đến *underfitting*.
- Lựa chọn k tối ưu thường dựa trên phương pháp kiểm định chéo (*cross-validation*).

3.3 Quy tắc phân loại mô hình

3.3.1 Chọn k láng giềng gần nhất

Sắp xếp các điểm huấn luyện theo khoảng cách tăng dần đến điểm mới x, sau đó lấy ra k điểm gần nhất để tạo thành tập $N_k(x)$.

3.3.2 Quy tắc phân loại

- **Đa số phiếu bầu**:
$$y = \arg\max_{C_j} \sum_{(x_i,y_i)\in N_k(x)} I(y_i = C_j)$$
- **Phiếu bầu có trọng số** (tùy chọn):
$$y = \arg\max_{C_j} \sum_{(x_i,y_i)\in N_k(x)} w_i \cdot I(y_i = C_j)$$
với trọng số $w_i = \frac{1}{d(x,x_i)}$ để ưu tiên các điểm gần hơn.

3.3.3 Công thức dự đoán

Xác suất một điểm thuộc lớp C_j được tính như sau:

$$P(C_j) = \frac{N_j}{k}$$

trong đó N_j là số láng giềng trong $N_k(x)$ thuộc lớp C_j. Nhãn dự đoán là:

$$\hat{y} = \arg\max_{C_j} P(C_j)$$

4 Mô hình Naive Bayes

4.1 Giới thiệu mô hình

Naive Bayes Classifier là một thuật toán phân loại đơn giản nhưng mạnh mẽ, đặc biệt phổ biến trong lĩnh vực học máy và xử lý ngôn ngữ tự nhiên. Mô hình này được xây dựng dựa trên Định lý Bayes—một công thức trong xác suất giúp tính toán khả năng xảy ra của một sự kiện khi đã biết thông tin từ trước. Với giả định rằng các đặc trưng (features) của dữ liệu độc lập với nhau khi biết lớp của dữ liệu, Naive Bayes trở thành một trong những thuật toán dễ hiểu và dễ triển khai nhất.

Ứng dụng thực tiễn của Naive Bayes rất đa dạng, từ phân loại thư rác (nhận diện và lọc bỏ email spam) đến phân tích cảm xúc (đánh giá cảm xúc trong các bình luận hoặc đánh giá trực tuyến). Khả năng xử lý nhanh chóng và hiệu quả trên cả dữ liệu nhỏ khiến nó trở thành lựa chọn phù hợp cho các tác vụ phân loại văn bản và các bài toán xử lý dữ liệu có chiều cao.

Thuật toán Naive Bayes là một phương pháp phân loại dựa trên Định lý Bayes với giả định rằng các đặc trưng (features) là độc lập với nhau. "Naive" nghĩa là "ngây thơ" vì nó giả định tính độc lập của các đặc trưng — giả định này không phải lúc nào cũng đúng, nhưng thường mang lại hiệu quả tốt trong nhiều ứng dụng thực tế.

4.2 Xây dựng Naive Bayes Classifier

Trong bài toán phân loại, chúng ta có C phân lớp (class) $1, 2, ..., C$. Với một điểm dữ liệu $x \in R^d$ (tức là một vector đầu vào x có d chiều), mục tiêu là xác định xác suất để x thuộc về mỗi class $k \in 1, 2, \ldots, C$. Biểu thức này có dạng:

$$p(y = k \mid x) = p(k \mid x)$$

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

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

Biểu thức $p(k \mid x)$ thường rất khó tính toán trực tiếp. Vì vậy, quy tắc Bayes được áp dụng để chuyển biểu thức về dạng dễ tính hơn:

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

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

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

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.
- $p(x \mid k)$: Phân phối xác suất của dữ liệu bên trong lớp k (xác suất để tồn tại một điểm dữ liệu x với điều kiện nó thuộc lớp k). Đây là một giá trị khó tính toán vì x có thể có nhiều chiều (nhiều đặc trưng).

Để đơn giản hóa việc tính toán, chúng ta giả định rằng các thành phần của x là độc lập với nhau nếu biết lớp k. Điều này dẫn đến biểu thức:

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

Giả định này, dù đơn giản và không thực tế, nhưng mang lại kết quả tốt trong nhiều ứng dụng.

Trong quá trình huấn luyện:

- Ước lượng $p(k)$ từ tần suất xuất hiện của mỗi lớp trong tập huấn luyện.
- Xác định $p(x_i \mid k)$ cho từng đặc trưng x_i dựa vào loại dữ liệu (có thể là Gaussian, Multinomial, hoặc Bernoulli).

Trong quá trình kiểm tra, với một điểm dữ liệu mới x, lớp của nó được xác định qua:

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

Để tránh sai số do tính toán với các xác suất nhỏ, ta thường dùng logarit của biểu thức trên:

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

Việc này không ảnh hưởng tới kết quả vì log là một hàm đồng biến trên tập các số dương.

4.3 Các phân phối thường dùng cho $p(x_i \mid k)$

- **Gausian Naive Bayes**

  Áp dụng cho dữ liệu liên tục, với giả định mỗi đặc trưng x_i trong lớp c có phân phối chuẩn. Xác suất có điều kiện $p(x_i \mid c)$ được tính như sau:

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

  trong đó:

  - $\mu_{c,i}$: Giá trị trung bình của đặc trưng x_i trong lớp c.
  - $\sigma_{c,i}^2$: Độ lệch chuẩn của đặc trưng x_i trong lớp c.

- **Multinomial Naive Bayes** Phù hợp cho dữ liệu rời rạc, đặc biệt là khi đặc trưng là số lần xuất hiện (ví dụ: tần suất từ). Xác suất có điều kiện $p(x_i \mid c)$ được tính dựa trên tần suất từ:

$$p(x_i \mid c) = \frac{N_{c,i} + \alpha}{N_c + \alpha d},$$

  trong đó:

  - $N_{c,i}$: Số lần đặc trưng x_i xuất hiện trong lớp c.
  - N_c: Tổng số lần xuất hiện của tất cả các đặc trưng trong lớp c.
  - d: Số lượng đặc trưng (từ vựng).
  - α: Tham số điều chỉnh (*smoothing parameter*), thường dùng Laplace smoothing với $\alpha = 1$.

- **Bernoulli Naive Bayes** Áp dụng cho dữ liệu nhị phân (có hoặc không có từ nào đó). Mỗi đặc trưng x_i chỉ có hai trạng thái (0 hoặc 1). Xác suất có điều kiện $p(x_i \mid c)$ được tính như sau:

$$p(x_i \mid c) = p(x_i = 1 \mid c)x_i + \big(1 - p(x_i = 1 \mid c)\big)(1 - x_i),$$

  trong đó:

  - $p(x_i = 1 \mid c)$: Xác suất để đặc trưng x_i xuất hiện (tức là $x_i = 1$) trong các mẫu thuộc lớp c.
  - x_i: Giá trị của đặc trưng x_i trong một mẫu cụ thể, có thể là 1 (nếu đặc trưng xuất hiện) hoặc 0 (nếu đặc trưng không xuất hiện).

Trong bài toán nhận diện bảng chữ cái ngôn ngữ ký hiệu American Sign Language (ASL), dữ liệu được xử lý như sau:

- **Dữ liệu đầu vào**: Hình ảnh 2D của các ký hiệu ASL (từ A đến Z) được chuyển thành một vector các pixel.
- **Loại dữ liệu**: Giá trị pixel trong ảnh grayscale là liên tục (continuous) hoặc gần liên tục sau khi chuẩn hóa về khoảng $[0, 1]$.

Kết luận: Dữ liệu pixel trong cách tiếp cận này là liên tục, do đó cần một biến thể Naive Bayes phù hợp với dữ liệu liên tục, chẳng hạn như *Gaussian Naive Bayes*.

5 Mô hình Multinomial Logistic Regression

5.1 Giới thiệu mô hình

Softmax Regression, còn được gọi là **Multinomial Logistic Regression**, là một phương pháp phân loại tuyến tính phổ biến trong học máy, đặc biệt được sử dụng cho các bài toán *phân loại đa lớp* (multi-class classification). Khác với hồi quy logistic truyền thống vốn chỉ xử lý các bài toán nhị phân (2 lớp), Softmax Regression cho phép mở rộng lên nhiều lớp. Mỗi giá trị đầu ra thể hiện **xác suất** của dữ liệu thuộc về một lớp cụ thể, từ đó mô hình đưa ra quyết định phân loại cuối cùng bằng cách chọn lớp có xác suất cao nhất.

Mô hình này được xây dựng trên cơ sở các **hàm tuyến tính** và sử dụng **hàm softmax** để chuẩn hóa đầu ra thành phân phối xác suất. Hàm softmax không chỉ đảm bảo các xác suất là số dương và có tổng bằng 1, mà còn có tính chất đồng biến — tức giá trị đầu ra lớn hơn sẽ tương ứng với xác suất cao hơn.

Softmax Regression được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau, tiêu biểu như: *phân loại chữ số viết tay* (ví dụ như bộ dữ liệu MNIST), *phân loại hình ảnh*, *nhận diện khuôn mặt*, và *phân tích văn bản* (như phân loại chủ đề hoặc cảm xúc trong câu).

Mặc dù là một mô hình tuyến tính, Softmax Regression có thể đạt hiệu quả cao trên các bài toán mà các lớp dữ liệu tách tuyến tính hoặc gần tuyến tính. Ngoài ra, nhờ vào cấu trúc đơn giản, nó có tốc độ huấn luyện nhanh và ít yêu cầu tài nguyên, vì vậy đây là lựa chọn lý tưởng trong các ứng dụng có giới hạn về thời gian hoặc thiết bị tính toán.

Với khả năng chuyển đổi đầu ra sang dạng xác suất và tích hợp tự nhiên với hàm mất mát cross-entropy, Softmax Regression không chỉ dễ triển khai mà còn có thể mở rộng linh hoạt trong các hệ thống học sâu, nơi nó thường được sử dụng như lớp đầu ra cuối cùng trong mạng nơ-ron phân loại.

5.2 Hàm softmax

Chúng ta cần một mô hình xác suất sao cho với mỗi input $\mathbf{x}$, a_i thể hiện xác suất để input đó rơi vào class i. Vậy điều kiện cần là các a_i phải dương và tổng của chúng bằng 1. Để có thể thỏa mãn điều kiện này, chúng ta cần *nhìn vào* mọi giá trị z_i và dựa trên quan hệ giữa các z_i này để tính toán giá trị của a_i.

Ngoài các điều kiện a_i lớn hơn 0 và có tổng bằng 1, chúng ta sẽ thêm một điều kiện nữa, đó là: giá trị $z_i = \mathbf{w}_i^\top \mathbf{x}$ càng lớn thì xác suất dữ liệu rơi vào class i càng cao. Điều kiện cuối này chỉ ra rằng chúng ta cần một hàm đồng biến đối với z_i.

Chú ý rằng z_i có thể nhận giá trị cả âm và dương. Một hàm số *mượt* đơn giản có thể chắc chắn biến z_i thành một giá trị dương, và hơn nữa, đồng biến, là hàm $\exp(z_i) = e^{z_i}$. Điều kiện *mượt* để thuận lợi hơn trong việc tính đạo hàm sau này. Điều kiện cuối cùng, tổng các a_i bằng 1 có thể được đảm bảo nếu:

$$a_i = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)}, \quad \forall i = 1, 2, \ldots, C$$

Hàm số này, tính trên tất cả các z_i, thỏa mãn tất cả các điều kiện đã xét: dương, tổng bằng 1, giữ được *thứ tự* của các z_i. Hàm số này được gọi là **softmax function**. Chú ý rằng với cách định nghĩa này, không có xác suất a_i nào tuyệt đối bằng 0 hoặc tuyệt đối bằng 1, mặc dù chúng có thể rất gần 0 hoặc 1 khi rất nhỏ hoặc rất lớn so sánh với các z_j, $j \neq i$.

Lúc này, ta có thể giả sử rằng:

$$P(y_k = i \mid \mathbf{x}_k; \mathbf{W}) = a_i$$

Trong đó, $P(y = i \mid \mathbf{x}; \mathbf{W})$ được hiểu là xác suất để một điểm dữ liệu $\mathbf{x}$ rơi vào class thứ i nếu biết tham số mô hình (ma trận trọng số) là $\mathbf{W}$.

5.3 Phiên bản ổn định hơn của softmax function

Khi một trong các z_i quá lớn, việc tính toán $\exp(z_i)$ có thể gây ra hiện tượng tràn số (overflow), ảnh hưởng lớn tới kết quả của hàm softmax. Có một cách khắc phục hiện tượng này bằng cách dựa trên quan sát sau:

$$\frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)} = \frac{\exp(-c)\exp(z_i)}{\exp(-c)\sum_{j=1}^{C} \exp(z_j)} = \frac{\exp(z_i - c)}{\sum_{j=1}^{C} \exp(z_j - c)}$$

với c là một hằng số bất kỳ.

5.4 One hot coding

Với cách biểu diễn như trên, mỗi output sẽ không còn là một giá trị tương ứng với mỗi class nữa mà sẽ là một vector có đúng 1 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 (sự thật là như thế, không cần dự đoán).

Khi sử dụng mô hình Softmax Regression, với mỗi đầu vào $\mathbf{x}$, ta sẽ có đầu ra dự đoán là $\mathbf{a} = \text{softmax}(\mathbf{W}^T\mathbf{x})$. Trong khi đó, đầu ra thực sự chúng ta có là vector $\mathbf{y}$ được biểu diễn dưới dạng one-hot coding.

Hàm mất mát sẽ được xây dựng để tối thiểu sự khác nhau giữa đầu ra dự đoán $\mathbf{a}$ và đầu ra thực sự $\mathbf{y}$. Một lựa chọn đầu tiên ta có thể nghĩ tới là:

$$J(\mathbf{W}) = \sum_{i=1}^{N} ||\mathbf{a}_i - \mathbf{y}_i||_2^2$$

Tuy nhiên đây chưa phải là một lựa chọn tốt. Khi đánh giá sự khác nhau (hay khoảng cách) giữa hai phân bố xác suất (probability distributions), chúng ta có một đại lượng để đo đếm khác hiệu quả hơn. Đại lượng đó có tên là cross entropy.

5.5 Cross Entropy

Cross entropy giữa hai phân phối p và q được định nghĩa là:

$$H(p, q) = E_p[-\log q]$$

Với p và q là rời rạc (như $\mathbf{y}$ và $\mathbf{a}$ trong bài toán của chúng ta), công thức này được viết dưới dạng:

$$H(p, q) = -\sum_{i=1}^{C} p_i \log q_i \quad (1)$$

Chú ý: Hàm cross entropy không tính đối xứng $H(p, q) \neq H(q, p)$. Điều này có thể dễ dàng nhận ra ở việc các thành phần của p trong công thức (1) có thể nhận giá trị bằng 0, trong khi đó các thành phần của q phải là dương vì $\log 0$ không xác định. Chính vì vậy, khi sử dụng cross entropy trong các bài toán học giám sát, p thường là đầu ra thực sự vì đầu ra thực sự chỉ có 1 thành phần bằng 1, còn lại bằng 0 (one-hot), q thường là đầu ra dự đoán, khi mà không có xác suất nào tuyệt đối bằng 1 hoặc tuyệt đối bằng 0 cả.

Với Softmax Regression, trong trường hợp có C classes, loss giữa đầu ra dự đoán và đầu ra thực sự của một điểm dữ liệu $\mathbf{x}_i$ được tính bằng:

$$J(\mathbf{W}; \mathbf{x}_i, \mathbf{y}_i) = -\sum_{j=1}^{C} y_{ij} \log(a_i)_j$$

Với y_{ij} và $(a_i)_j$ lần lượt là phần tử thứ j của vector (xác suất) $\mathbf{y}_i$ và $\mathbf{a}_i$. Nhắc lại rằng đầu ra $\mathbf{a}_i$ phụ thuộc vào đầu vào $\mathbf{x}_i$ và ma trận trọng số $\mathbf{W}$.

5.6 Hàm mất mát cho Softmax Regression

Kết hợp tất cả các cặp dữ liệu $(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, ..., N$, chúng ta sẽ có hàm mất mát cho Softmax Regression như sau:

$$J(\mathbf{W}; \mathbf{X}, \mathbf{Y}) = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \log(a_i)_j = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \log\left(\frac{\exp(\mathbf{w}_j^T \mathbf{x}_i)}{\sum_{k=1}^{C} \exp(\mathbf{w}_k^T \mathbf{x}_i)}\right)$$

Với ma trận trọng số $\mathbf{W}$ là biến cần tối ưu. Hàm mất mát này trông có vẻ đáng sợ, nhưng đừng sợ, đọc tiếp các bạn sẽ thấy đạo hàm của nó rất đẹp (và đáng yêu).

5.7 Tối ưu hàm mất mát

Một lần nữa, chúng ta lại sử dụng Stochastic Gradient Descent (SGD) ở đây.

Với mỗi cặp dữ liệu $(\mathbf{x}_i, \mathbf{y}_i)$, ta có:

$$
\begin{aligned}
&J_i(\mathbf{W})J(\mathbf{W};\mathbf{x}_i,\mathbf{y}_i) \\
&\quad = -\sum_{j=1}^{C} y_{ij} \log\left(\frac{\exp(\mathbf{w}_j^T\mathbf{x}_i)}{\sum_{k=1}^{C}\exp(\mathbf{w}_k^T\mathbf{x}_i)}\right) \\
&\quad = -\sum_{j=1}^{C} y_{ij}\mathbf{w}_j^T\mathbf{x}_i + \sum_{j=1}^{C} y_{ij}\log\left(\sum_{k=1}^{C}\exp(\mathbf{w}_k^T\mathbf{x}_i)\right) \quad (3)
\end{aligned}
$$

Trong biến đổi ở dòng cuối cùng, tôi đã sử dụng quan sát: $\sum_{j=1}^{C} y_{ij} = 1$ vì nó là tổng các xác suất.

Tiếp theo ta sử dụng công thức:

$$
\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} = \left[\frac{\partial J(\mathbf{W})}{\partial \mathbf{w}_1}, \frac{\partial J(\mathbf{W})}{\partial \mathbf{w}_2}, ..., \frac{\partial J(\mathbf{W})}{\partial \mathbf{w}_C}\right] \quad (4)
$$

Trong đó, gradient theo từng được dựa theo dựa theo (3):

$$
\begin{aligned}
\frac{\partial J_i(\mathbf{W})}{\partial \mathbf{w}_j} &= -y_{ij}\mathbf{x}_i + \frac{\exp(\mathbf{w}_j^T\mathbf{x}_i)}{\sum_{k=1}^{C}\exp(\mathbf{w}_k^T\mathbf{x}_i)}\mathbf{x}_i \\
&= -y_{ij}\mathbf{x}_i + (a_i)_j\mathbf{x}_i \\
&= ((a_i)_j - y_{ij})\mathbf{x}_i e_{ij}\mathbf{x}_i \quad (5)
\end{aligned}
$$

Giá trị $e_{ij} = (a_i)_j - y_{ij}$ có thể được coi là sai số dự đoán.

Đến đây ta đã có biểu thức thực sự rất đẹp. Kết hợp (4) và (5) ta có:

$$
\frac{\partial J_i(\mathbf{W})}{\partial \mathbf{W}} = [e_{i1}\mathbf{x}_i, e_{i2}\mathbf{x}_i, ..., e_{iC}\mathbf{x}_i] = \mathbf{x}_i\mathbf{e}_i^T
$$

với $\mathbf{E} = \mathbf{A} - \mathbf{Y}$. Công thức tính gradient đơn giản này giúp cho cả Batch Gradient Descent, Stochastic Gradient Descent (SGD), và Mini-batch Gradient Descent đều có thể dễ dàng được áp dụng.

Giả sử rằng chúng ta sử dụng SGD, công thức cập nhật cho ma trận trọng số $\mathbf{W}$ sẽ là:

$$
\mathbf{W} = \mathbf{W} + \eta\mathbf{x}_i(\mathbf{y}_i - \mathbf{a}_i)^T
$$

6 Thực nghiệm

6.1 Kết quả các chỉ số của ba mô hình: Naive Bayes, Multinomial Logistic Regression, KNN

Ta sẽ so sánh độ chính xác hai mô hình thông qua các chỉ số *accuracy*, *precision*, *recall*, *confusion matrix*.

Các chỉ số của mô hình Gaussian Naive Bayes

Các chỉ số hiệu suất:

- *Train time*: 0.0316 giây,
- *Accuracy*: 0.5939,
- *Precision*: 0.6616,
- *Recall*: 0.5939.

Các chỉ số của mô hình Multinomial Logistic Regression

Các chỉ số hiệu suất:

- *Train time*: 1.4367 giây,
- *Accuracy*: 0.7680,
- *Precision*: 0.7706,
- *Recall*: 0.7680.

Các chỉ số của mô hình KNN

Các chỉ số hiệu suất:

- *Train time*: 0.0147 giây,
- *Accuracy*: 0.9928,
- *Precision*: 0.9928,
- *Recall*: 0.9928.

Nội dung so sánh: Các chỉ số hiệu suất được tính toán trên tập kiểm tra sau khi huấn luyện từng mô hình trên vector pixel đã qua tiền xử lý (resize, chuẩn hóa và PCA).

Bảng so sánh các chỉ số hiệu suất:

Mô hình	Train time (giây)	Accuracy	Precision	Recall
Gaussian Naive Bayes	0.0316	0.5939	0.6616	0.5939
Multinomial Logistic Regression	1.4367	0.7680	0.7706	0.7680
KNN	0.0147	0.9928	0.9928	0.9928

Phân tích kết quả:

- *Thời gian huấn luyện*: KNN có thời gian huấn luyện nhanh nhất (0.0147 giây), tiếp theo là Gaussian Naive Bayes (0.0316 giây), trong khi Multinomial Logistic Regression chậm nhất (1.4367 giây).
- *Độ chính xác (Accuracy)*: KNN đạt độ chính xác cao nhất (99.28%), vượt trội so với Multinomial Logistic Regression (76.80%) và Gaussian Naive Bayes (59.39%).
- *Độ chính xác từng lớp (Precision) và độ phủ (Recall)*: KNN có giá trị cao nhất và cân bằng (99.28% cho cả hai), tiếp theo là Multinomial Logistic Regression (77.06% và 76.80%), trong khi Gaussian Naive Bayes thấp nhất (66.16% và 59.39%).

7 Kết luận

Nhìn chung, trong bài toán nhận diện ngôn ngữ ký hiệu, mô hình **KNN** nổi bật là lựa chọn tối ưu nhất nhờ vào độ chính xác vượt trội, khả năng dự đoán cân bằng và thời gian huấn luyện nhanh chóng, đáp ứng tốt yêu cầu của bài toán. So sánh với **Multinomial Logistic Regression**, mô hình này đạt hiệu suất khá, phù hợp cho các tình huống cần cân bằng giữa độ chính xác và độ phức tạp tính toán, nhưng vẫn không thể sánh bằng KNN về hiệu quả tổng thể. Trong khi đó, **Gaussian Naive Bayes** thể hiện hiệu suất kém nhất, cho thấy không phù hợp với đặc điểm của dữ liệu hình ảnh đã qua tiền xử lý trong bài toán này.

Để nâng cao hiệu quả của các mô hình, một số cải tiến có thể được áp dụng:

- Đối với KNN, nên tập trung vào tối ưu hóa bằng cách sử dụng các kỹ thuật giảm chiều dữ liệu tiên tiến như t-SNE hoặc UMAP, đồng thời áp dụng các cấu trúc dữ liệu như KD-Tree để tăng tốc độ tìm kiếm láng giềng, đặc biệt khi xử lý dữ liệu lớn.
- Với Multinomial Logistic Regression, việc tinh chỉnh siêu tham số thông qua Grid Search và bổ sung các đặc trưng bổ trợ (như đặc trưng hình học) có thể cải thiện đáng kể hiệu suất.
- Đối với Gaussian Naive Bayes, thử nghiệm các biến thể khác như Multinomial Naive Bayes hoặc cải thiện quy trình tiền xử lý dữ liệu, chẳng hạn như cân bằng lớp, sẽ giúp tăng hiệu quả.
- Các cải tiến chung như sử dụng kỹ thuật tăng cường dữ liệu (*data augmentation*), thử nghiệm các mô hình học sâu như Convolutional Neural Networks (CNN) nếu có đủ tài nguyên, và kiểm tra hiệu suất trên dữ liệu thực tế (như video ký hiệu) sẽ đảm bảo hệ thống hoạt động ổn định và hiệu quả hơn trong các ứng dụng thực tiễn, từ đó nâng cao chất lượng nhận diện ngôn ngữ ký hiệu.