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

***

BÁO CÁO TIỂU LUẬN GIỮA KỲ

CHỦ ĐỀ 10: NHẬN DẠNG HÌNH ẢNH CHỮ SỐ VIẾT TAY

Học phần : **Học máy**

Mã học phần: **MAT3533**

Giảng viên phụ trách: **Cao Văn Chung**

Hà Nội, 2024

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

———————***———————

BÁO CÁO TIỂU LUẬN GIỮA KỲ

CHỦ ĐỀ 10: NHẬN DẠNG HÌNH ẢNH CHỮ SỐ VIẾT TAY

Học phần :	**Học máy**
Mã học phần:	**MAT3533**
Giảng viên phụ trách:	**Cao Văn Chung**

Hà Nội, 2024

LỜI NÓI ĐẦU

Trong thời đại số hóa ngày nay, việc tự động nhận dạng chữ số từ hình ảnh chữ viết tay đã trở thành một phần quan trọng của cuộc sống hiện đại với nhiều ứng dụng trong công nghệ thông tin và trí tuệ nhân tạo. Vấn đề này đặt ra những thách thức và cơ hội đối với cả con người và máy móc. Tính ứng dụng của việc nhận dạng chữ viết tay đã lan rộng ra nhiều lĩnh vực khác nhau trong thế giới thực, từ viết nhãn tự động trên bưu kiện, quét mã vạch tại các cửa hàng, cho đến nhận dạng chữ ký số trong giao dịch điện tử và công nghệ nhận dạng biển số xe. Trong học tập và nghiên cứu, việc tự động nhận dạng chữ số từ hình ảnh chữ viết tay giúp tạo ra các bộ dữ liệu lớn và đa dạng, từ đó cung cấp cơ sở cho việc phát triển và kiểm định các thuật toán và mô hình học máy mới. Tuy nhiên, đây vẫn là một thách thức lớn do tính đa dạng và biến thể của chữ viết tay cũng như các vấn đề về nhiễu, biến dạng bởi các yếu tố như độ phân giải thấp, ánh sáng không đồng đều, góc chụp trong quá trình nhận dạng hình ảnh.

Trong bối cảnh ấy và trong khuôn khổ của học phần Học máy, nhóm em quyết định lựa chọn chủ đề 10, tập trung nhận dạng chữ viết tay thông qua bộ dữ liệu MNIST bằng hai mô hình: Naive Bayes, Artificial Neural Networks. Sau đó, thực hiện so sánh kết quả thực nghiệm của hai mô hình để có được một cái nhìn tổng quan về ưu và nhược điểm của từng mô hình.

Mặc dù nhóm đã rất cố gắng trong quá trình thực hiện, song bài báo cáo sẽ không tránh khỏi những thiếu sót do còn hạn chế về mặt kiến thức. Vì vậy, nhóm em luôn sẵn lòng đón nhận những nhận xét, đánh giá và rất mong nhận được những ý kiến đóng góp của thầy và các bạn để có thể hoàn thiện hơn.

Nhóm em xin được bày tỏ sự cảm ơn chân thành tới thầy Cao Văn Chung đã có sự hướng dẫn và tin tưởng để chúng em có thể hoàn thành đề tài này. Đồng thời, nhóm em cũng xin được bày tỏ sự biết ơn tới những nhận xét, ý kiến đóng góp từ thầy và các bạn giúp nhóm hoàn thiện hơn trong đề tài.

Hà Nội, ngày 28 tháng 04 năm 2024

Nhóm thực hiện

THÀNH VIÊN NHÓM

Thành viên 1:

Họ và tên: Vũ Nguyễn Huy Hoàng

Mã sinh viên: 21000505

Lớp chuyên ngành: K66A2 Toán tin

Thành viên 2:

Họ và tên: Nguyễn Thị Thanh Hương

Mã sinh viên: 21000506

Lớp chuyên ngành: K66A2 Toán tin

Thành viên 3:

Họ và tên: Đặng Ngọc Quân

Mã sinh viên: 21000699

Lớp chuyên ngành: K66A2 Toán tin

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

DANH MỤC CÁC CHỮ VIẾT TẮT

STT	**Ký hiệu**	**Nguyên nghĩa**
2	ANN	Mạng nơ-ron nhân tạo
1	MNIST	Bộ dữ liệu chứa hình ảnh các chữ số viết tay từ 0 đến 9
3	PCA	Phân tích thành phần chính

MỞ ĐẦU

1 Mục tiêu nghiên cứu

Bài tiểu luận này tập trung vào phân tích và ứng dụng các mô hình học máy gồm Naïve Bayes, Artificial Neural Networks (ANN) để nhận dạng chữ số viết tay từ hình ảnh. Nhóm sẽ tiến hành thử nghiệm, sau đó đánh giá và so sánh các mô hình thông qua các chỉ số: accuracy, confusion matrix, recall và precision.

2 Phương pháp nghiên cứu

Để tiến hành đề tài này, nhóm em sẽ thực hiện theo quy trình sau:

- Đầu tiên, thực hiện các phương pháp tiền xử lý dữ liệu và trực quan hóa các phân lớp dữ liệu dưới dạng 2D để hiểu rõ hơn về phân bố của dữ liệu.
- Tiếp đó, tiến hành xây dựng và huấn luyện mô hình Naive Bayes và mô hình ANN trên tập dữ liệu training được rút gọn về 100 chiều.
- Cuối cùng, dựa trên các các chỉ số: accuracy, confusion matrix, recall, precision so sánh hai mô hình đã thử nghiệm rồi đưa ra kết luận về ưu và nhược điểm của từng mô hình.

3 Đối tượng và phạm vi nghiên cứu

Báo cáo sẽ tiến hành thử nghiệm trên bộ dữ liệu MNIST bằng hai mô hình học máy: Naive Bayes, ANN thông qua các thư viện sẵn có trên python: NumPy, Pandas, Matplotlib, Scikit-learn, TensorFlow, Keras.

4 Cấu trúc báo cáo

Nội dung bài báo cáo bao gồm 4 chương. Cụ thể:

Chương I: Giới thiệu chung về bộ dữ liệu, bao gồm nguồn gốc, kích thước, cấu trúc bộ dữ liệu

Chương II: Trình bày kiến thức cơ bản về phương pháp tiền xử lý dữ liệu, bao gồm: cách chuẩn hóa dữ liệu (Standardization) và cách giảm chiều dữ liệu (Principal Component Analysis).

Chương III: Giới thiệu chung về mô hình Naive Bayes

Chương IV: Giới thiệu chung về mô hình ANN

Chương V: Tiến hành thực nghiệm các mô hình trên bộ dữ liệu. Sau đó, đánh giá và so sánh các mô hình.

CHƯƠNG I: GIỚI THIỆU BỘ DỮ LIỆU

1.1 Tổng quan về bộ dữ liệu MNIST

Tập dữ liệu MNIST là một trong những tập dữ liệu phổ biến nhất trong lĩnh vực nhận dạng chữ số viết tay. Nó được tạo ra từ cơ sở dữ liệu của Viện Tiêu chuẩn và Công nghệ Quốc gia của Hoa Kỳ. Tuy nhiên, MNIST đã được sửa đổi để phù hợp với các mô hình máy học và trở thành một tập dữ liệu tiêu chuẩn cho các bài toán nhận dạng chữ số.

1.2 Nguồn dữ liệu

Đã được giảng viên cung cấp trong các bài thực hành

1.3 Cấu trúc của bộ dữ liệu

Tập dữ liệu MNIST bao gồm hai phần chính:

- Tập dữ liệu huấn luyện: Gồm 60.000 hình ảnh chữ số viết tay từ 0 đến 9 bởi các người tham gia khác nhau và có độ biến thể nhất định, mỗi hình ảnh có kích thước 28x28 pixel.
- Tập dữ liệu kiểm tra: Gồm 10.000 hình ảnh chữ số viết tay, cũng có kích thước 28x28 pixel.

Mỗi hình ảnh trong tập dữ liệu MNIST là một ảnh xám, có nghĩa là mỗi pixel chỉ chứa một giá trị từ 0 đến 255 đại diện cho độ sáng của pixel tương ứng. Giá trị càng cao, pixel càng sáng. Các giá trị này được sử dụng để hiển thị cường độ sáng của mỗi pixel trong hình ảnh.

CHƯƠNG II: PHƯƠNG PHÁP TIỀN XỬ LÝ DỮ LIỆU

2.1 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.

2.1.1 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.

2.1.2 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:

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

Trong đó: mean(X) là kỳ vọng của X, std(X) là phương sai của X

Ví dụ: Với tập dữ liệu X có $mean(X) = 10$, $std(X) = 4$, giá trị $x = 15.9$ sẽ được chuẩn hóa thành: $x' = (15.9 - 10)/4 = 1.475$

2.1.3 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.

2.2 Giảm chiều dữ liệu

Giảm chiều dữ liệu là một kỹ thuật quan trọng giúp giảm số lượng đặc trưng trong dữ liệu mà vẫn giữ lại lượng thông tin quan trọng nhất. Trong các phương pháp giảm chiều dữ liệu, phân tích thành phần chính (PCA) thường được coi là phương pháp đơn giản và hiệu quả nhất cho nhiều ứng dụng.

2.2.1 Mục đích

PCA là một phương pháp giảm chiều tuyến tính có mục đích chính là giảm kích thước, giảm nhiễu dữ liệu, tối ưu hóa dung lượng lưu trữ, chuẩn bị dữ liệu cho các thuật toán học sâu hơn.

2.2.2 Phương pháp PCA

Nội dung phương pháp: Phương pháp này sẽ sử dụng phép biến đổi trực giao để biến đổi dữ liệu từ không gian nhiều chiều sang một không gian mới ít chiều hơn nhằm tối đa hóa phương sai dữ liệu của không gian mới đó. Sau đó, lựa chọn ra những chiều có phương sai lớn nhất.

Các bước thực hiện:

- **Bước 1: Chuẩn hóa dữ liệu**
  Trước tiên, chuẩn hóa dữ liệu bằng cách trừ đi giá trị trung bình của mỗi đặc trưng để dữ liệu có trung bình bằng 0: $X_{norm} = X - \bar{X}$
  Trong đó, $\bar{X} = \frac{1}{N}\sum_{n=1}^{N} x_n$ là vector trung bình của mỗi cột trong ma trận dữ liệu.

- **Bước 2: Tính ma trận hiệp phương sai**
  Ma trận hiệp phương sai S của dữ liệu được tính như sau: $S = \frac{1}{N}X_{norm}^T X_{norm}$
  Trong đó, N là số lượng mẫu dữ liệu. Các phần tử của ma trận hiệp phương sai biểu thị sự tương quan cộng tính giữa các cặp đặc trưng

- **Bước 3: Tính các giá trị riêng và vector riêng**
  Phân tích giá trị riêng của ma trận hiệp phương sai để tìm các vectơ riêng v_i tương ứng với giá trị riêng λ_i: $Sv_i = \lambda_i v_i$
  Các giá trị riêng λ_i đo lường mức độ biến thiên của dữ liệu dọc theo vectơ riêng tương ứng, và các vectơ riêng là các hướng mới mà dọc theo đó, dữ liệu có sự biến thiên lớn nhất.

- **Bước 4: Chọn các thành phần chính**
  Chọn k vectơ riêng có giá trị riêng lớn nhất, tương ứng với k thành phần chính có sự biến thiên lớn nhất: $V_k = [v_1, v_2, \ldots, v_k]$
  Các vectơ riêng này tạo thành cơ sở của không gian mới mà dữ liệu sẽ được chiếu xuống.

- **Bước 5: Chiếu dữ liệu xuống không gian mới** Chiếu dữ liệu từ không gian gốc xuống không gian kích thước k bằng cách nhân ma trận dữ liệu ban đầu (đã chuẩn hóa) với ma trận chứa k vectơ riêng: $Y = X_{norm}^{T} V_k$
  Trong đó, Y là biểu diễn của dữ liệu trong không gian mới với k chiều.

2.2.3 Tác động

- **Đối phó với độ phức tạp cao:** Khi làm việc số lượng lớn đặc trưng, mô hình học máy trở nên phức tạp hơn, gây khó khăn trong quá trình huấn luyện và tăng yêu cầu về tài nguyên tính toán và bộ nhớ. Giảm chiều dữ liệu giúp giảm bớt các đặc trưng mà vẫn giữ lại những thông tin quan trọng nhất, điều này giúp giảm bớt độ phức tạp của mô hình, tiết kiệm tài nguyên và làm cho việc huấn luyện trở nên dễ dàng hơn

- **Tránh overfitting:** Mô hình có thể trở nên quá phức tạp và có khả năng "học thuộc lòng" dữ liệu huấn luyện mà không khái quát hóa tốt trên dữ liệu mới. PCA giúp mô hình đỡ phức tạp hơn và giảm khả năng "học thuộc lòng" dữ liệu huấn luyện mà không khái quát hóa tốt trên dữ liệu mới.

- **Cải thiện độ chính xác:** Quá nhiều đặc trưng có thể ẩn chứa nhiễu và thông tin thừa làm giảm khả năng phát hiện các mối quan hệ quan trọng giữa các đặc trưng của mô hình. PCA có thể loại bỏ nhiễu và thông tin thừa trong dữ liệu giúp cải thiện độ chính xác và khả năng dự đoán của mô hình.

- **Giảm khó khăn trong việc giải thích mô hình:** Mô hình với nhiều đặc trưng có thể khó giải thích, làm giảm khả năng hiểu và truyền đạt cách mô hình hoạt động cho người khác. PCA giúp tạo ra các mô hình đơn giản và dễ giải thích hơn bằng cách giảm số lượng đặc trưng.

- **Quản lý dữ liệu:** Quản lý và xử lý dữ liệu với số chiều cao yêu cầu nhiều tài nguyên về lưu trữ và tính toán. PCA được coi như một phương pháp nén dữ liệu giúp giảm tải trọng lưu trữ và tài nguyên tính toán. Điều này làm cho quản lý và xử lý dữ liệu trở nên dễ dàng hơn và giảm thiểu chi phí lưu trữ và tính toán.

CHƯƠNG III: MÔ HÌNH NAIVE BAYES

3.1 Giới thiệu mô hình

Naive Bayes là mô hình học máy dựa trên định lý Bayes để dự đoán phân lớp $c \in \{1,2,\ldots,C\}$ của một điểm dữ liệu $x \in R^d$

Cụ thể, Naïve Bayes tính toán xác suất của một lớp dữ liệu (hay biến phụ thuộc) dựa trên các giá trị của các đặc trưng (biến độc lập) bằng công thức:

$$P(c|x) = \frac{P(x|c) * P(c)}{P(x)}$$

Trong đó:
$P(c|x)$ là xác suất có điều kiện của lớp c dựa trên dữ liệu x.
$P(x|c)$ là xác suất có điều kiện của dữ liệu x dưới điều kiện lớp c.
$P(c)$ là xác suất tiên nghiệm của lớp c, tức là xác suất mà một điểm dữ liệu bất kỳ thuộc vào lớp c mà không liên quan x.
$P(x)$ là xác suất của dữ liệu x, tức là xác suất tổng của dữ liệu x xuất hiện trong mọi lớp.

Trong thực tế, định lý Bayes thường được áp dụng như sau:

$$P(c|x) = P(x|c)P(c) \tag{1}$$

3.2 Quá trình huấn huyện, dự đoán:

Bước 1: Tính toán xác suất tiên nghiệm của từng lớp dữ liệu trong tập huấn luyện, tức là xác suất mà một điểm dữ liệu thuộc vào từng lớp.

$$P(y = c) = \frac{|\{x \in X | y(x) = c\}|}{|X|}$$

Trong đó: $|\{x \in X | y(x) = c\}|$ là số phần tử thuộc vào phân lớp c
$|X|$ là số phần tử của tập dữ liệu

Bước 2: Với mỗi đặc trưng, tính toán xác suất có điều kiện của giá trị đó dưới mỗi lớp dữ liệu.

$$P(x|c) = \prod_{i=1}^{d} P(x_i|c)$$

Bước 3: Sử dụng công thức (1) tính $P(c|x)$ để xác định khả năng x rơi vào mỗi lớp c dựa trên xác suất tiên nghiệm và xác suất có điều kiện đã tính được.
Sau đó, dự đoán đầu ra c cho x là lớp có xác suất lớn nhất:

$$c = \arg \max_{c \in \{1,\ldots,C\}} P(x|c)P(c)$$

3.3 Tính xác suất P(x|c)

Việc tính các xác suất $P(x_i|x)$ phụ thuộc vào phân phối của dữ liệu. Trong mô hình Naive Bayes, có ba phân phối thường được dùng để tính các $P(x_i|c)$:

- **Gaussian Naive Bayes**: dùng cho dữ liệu mà mỗi đặc trưng được giả định là phân phối chuẩn (phân phối Gaussian)
- **Multinomial Naive Bayes**: thường được sử dụng cho các bài toán phân loại văn bản
- **Bernoulli Naive Bayes**: được áp dụng cho các loại dữ liệu mà mỗi thành phần là một giá trị 0 hoặc 1

Đối với bài toán nhận dạng hình ảnh chữ số viết tay:
Trong bộ dữ liệu MNIST, mỗi hình ảnh chữ số được biểu diễn bằng các giá trị pixel có giá trị từ 0 đến 255, thể hiện độ sáng của mỗi pixel. Mỗi pixel này có thể được xem xét như một biến ngẫu nhiên tuân theo phân phối Gaussian với một giá trị kỳ vọng và độ lệch chuẩn. Như vậy, trong bài toán này, các xác suất $P(x_i|c)$ sẽ được tính theo công thức:

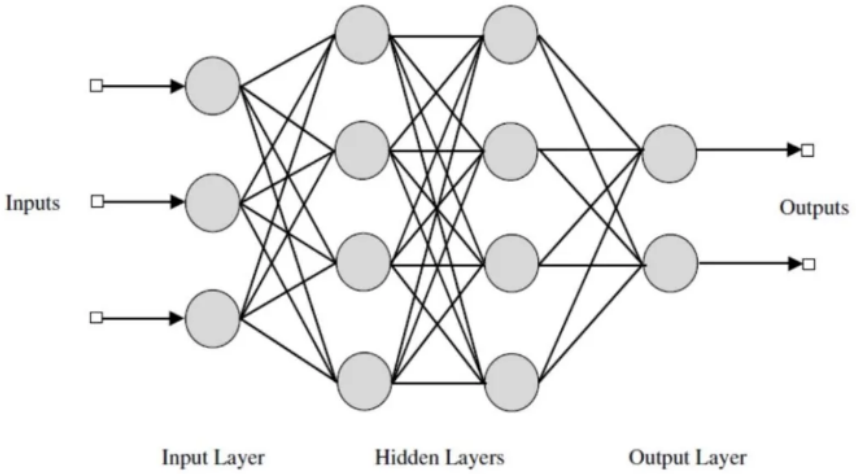
$$P(x_i|c) = \frac{1}{\sqrt{2\pi\sigma_{ci}^2}} \exp\left(-\frac{1}{2\sigma_{c_i}^2}(x_i - \mu_{ci})^2\right)$$

Trong đó: μ_{ci} là giá trị kỳ vọng của phân phối Gaussian cho đặc trưng x_i trong lớp c
σ_{ci} là độ lệch chuẩn của phân phối Gaussian cho đặc trưng x_i trong lớp c

CHƯƠNG IV: MÔ HÌNH ANN

4.1 Kiến trúc cơ bản

Một mạng nơ-ron nhân tạo truyền thẳng full connection có cấu trúc như sau:



Hình 4.1 Kiến trúc cơ bản của mạng ANN

Mạng bao gồm một số lượng lớn các nơ-ron nhân tạo được tổ chức thành từng lớp, mỗi nơ-ron trong mỗi lớp được kết nối với tất cả các nơ-ron ở lớp trước đó và sau đó thông qua các trọng số w. ANN thường có 3 lớp:

Lớp đầu vào (input layer): Là lớp nhận dữ liệu đầu vào, số lượng nơ-ron trong lớp tương ứng với số chiều của dữ liệu đầu vào.

Lớp ẩn (hidden layer): Là lớp chứa các nơ-ron ẩn giúp kết nối giá trị đầu vào đến giá trị đầu ra. Một mạng nơ-ron có thể có một hoặc nhiều lớp ẩn chịu trách nhiệm thực hiện các phép biến đổi trên dữ liệu đầu vào thông qua hàm kích hoạt tạo ra các biểu diễn phi tuyến của dữ liệu, giúp mô hình có khả năng học được các mối quan hệ phức tạp giữa các đặc trưng. Số lượng nơ-ron được tự điều chỉnh sao cho phù hợp.

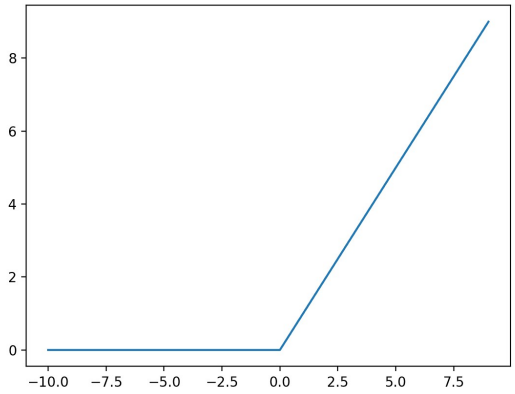
Lớp đầu ra (output layer): Là lớp cuối cùng của mạng, nơi các dự đoán hoặc kết quả được tạo ra. Số lượng nơ-ron bằng số lượng lớp phân loại.

4.2 Hàm kích hoạt ReLU

Hàm kích hoạt trong một mạng nơ-ron giống như một "công tắc" cho mỗi nơ-ron, nơi mà đầu ra của nơ-ron sẽ được điều chỉnh dựa trên đầu vào của nó. Thay vì chỉ đơn giản truyền đầu vào tới đầu ra, hàm kích hoạt sẽ thêm vào một lớp tính toán phi tuyến tính, giúp mạng nơ-ron học và biểu diễn các mối quan hệ phúc tạp trong dữ liệu.

Trong bài toán nhận dạng hình ảnh chữ số viết tay, hàm kích hoạt ReLU là một lựa chọn phổ biến và hiệu quả.

Công thức hàm ReLU: $f(z) = max(0, z)$



Hình 4.2 Đồ thị hàm kích hoạt ReLU

Nếu giá trị đầu vào z dương thì hàm sẽ giữ nguyên giá trị đó, nếu z âm hàm sẽ chuyển nó thành 0

4.3 Tối ưu hóa hàm chi phí bằng thuật toán Adam

Thuật toán tối ưu hóa là một kỹ thuật quan trọng trong việc xây dựng mô hình mạng ANN, đây là quy trình được lặp đi lặp lại so sánh các giải pháp khác nhau cho đến khi tìm được một giải pháp tối thiểu hóa hàm mất mát. Về cơ bản, thuật toán tối ưu giúp mạng ANN "học" từ dữ liệu đầu vào để tìm ra một tập các trọng số - w và bias - b phù hợp.

Ở đây, ta lựa chọn hàm mất mát cross entropy:

$$J(w,b) = -\sum_{i=1}^{N}(y_i \log(a_i^{(L)}) + (1-y_i)\log(1-a_i^{(L)}))$$

$a_i^{(L)}$ *là dự đoán đầu ra của* x_i *qua mô hình*

Thuật toán Adam sử dụng hai giá trị moment1 (m), moment2 (v) để lưu trữ các giá trị gradient và gradient bình phương của các bước trước đó. Các giá trị m, v được tính bằng công thức:

$$m_w^t = \beta_1 m_w^{t-1} + (1-\beta_1)d_w^t \quad , \quad z_w^t = \beta_2 v_w^{t-1} + (1-\beta_2)(d_w^t)^2$$
$$m_b^t = \beta_1 m_b^{t-1} + (1-\beta_1)d_b^t \quad , \quad z_b^t = \beta_2 v_b^{t-1} + (1-\beta_2)(d_b^t)^2$$

Trong đó: $d_w = \frac{\partial J(w,b)}{\partial w}$, $d_b = \frac{\partial J(w,b)}{\partial b}$

β_1, β_2 là các trọng số không âm, ta thường chọn $\beta_1 = 0.9$, $\beta_2 = 0.999$.

Nếu các giá trị khởi tạo m^t, v^t là các vector 0, các giá trị này sẽ có khuynh hướng nghiêng về 0, đặc biệt là khi β_1 và β_2 sấp sỉ bằng 1. Ta khắc phục điều này bằng cách ước lượng các giá trị như sau:

$$\hat{m}^t = \frac{m^t}{1-\beta_1^t} \quad , \quad \hat{v}^t = \frac{v^t}{1-\beta_2^t}$$

Sau đó, các trọng số - w và bias - b sẽ được cập nhật theo công thức:

$$w^{t+1} = w^t - \frac{\eta}{\sqrt{v_w^t + \varepsilon}} \hat{m_w}^t \qquad , \qquad b^{t+1} = b^t - \frac{\eta}{\sqrt{v_b^t + \varepsilon}} \hat{m_b}^t$$

4.4 Huấn luyện mô hình ANN với backpropagation

Thuật toán lan truyền ngược backpropagation có mục đích tính toán các gradient $\frac{\partial J(w,b)}{\partial w}$, $\frac{\partial J(w,b)}{\partial b}$ của hàm mất theo trọng số và bias.

Quá trình lan truyền tiến (feedforward): Dữ liệu huấn luyện được đưa qua các lớp của mạng nơ-ron để tính toán đầu ra dự đoán $\hat{y} = a^{(L)}$. Quá trình này được biểu diễn bằng công thức:

$$z_i^{(l)} = \sum_{j=1}^{n^l} w_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)} \tag{2}$$

$$a_i^{(l)} = f(z_i^{(l)}) \tag{3}$$

Trong đó: n^l là số neuron ở lớp l; z_i^l là đầu vào của neuron thứ i ở lớp l.
Đầu ra của neuron thứ i ở lớp l được biểu diễn bằng a_i^l ứng với hàm kích hoạt $f(z_i)$ tương ứng.

Quá trình lan truyền ngược (backpropagation): Đây quá trình tính ngược các đạo hàm từ lớp L - lớp cuối của mạng neural tới lớp 1 theo công thức:

$$\frac{\partial J}{\partial w_{ij}^{(l)}} = \frac{\partial J}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}} = \frac{\partial J}{\partial a_j^{(l)}} a_i^{(l-1)} f'(z_j^{(l)})$$

$$\frac{\partial J}{\partial b_j^{(l)}} = \frac{\partial J}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial b_j^{(l)}} = \frac{\partial J}{\partial a_j^{(l)}} f'(z_j^{(l)})$$

Với việc sử dụng hàm kích hoạt ReLU và hàm mất mát cross entropy, ta có:

$$\frac{\partial J}{\partial a_j^{(l)}} = \frac{-y_j}{a_j^{(l)}} + \frac{1 - y_j}{1 - a_j^{(l)}}$$

$$f'(z) = \begin{cases} 1, z > 0 \\ 0, z \leq 0 \end{cases}$$

Cuối cùng, ta sử dụng các gradient đã tính được để cập nhật các trọng số (w) và bias (b) bằng thuật toán Adam.

CHƯƠNG V: THỰC NGHIỆM

5.1 Chuẩn bị dữ liệu

5.1.1 Đọc dữ liệu

Đầu tiên, xây dựng code để tải dữ liệu hình ảnh và nhãn tương ứng từ các tệp gzip:

```
import os
import numpy as np
import gzip

def get_mnist_data(images_path, labels_path, num_images,
                   shuffle=False, _is=True, image_size=28):
    f_images = gzip.open(images_path,'r')
    f_images.read(16)
    real_num = num_images if not shuffle else (60000 if _is else 10000)
    buf_images = f_images.read(image_size * image_size * real_num)
    images = np.frombuffer(buf_images, dtype=np.uint8).astype(np.float32)
    images = images.reshape(real_num, image_size, image_size,)
    f_labels = gzip.open(labels_path,'r')
    f_labels.read(8)
    labels = np.zeros((real_num)).astype(np.int64)
    for i in range(0, real_num):
        buf_labels = f_labels.read(1)
        labels[i] = np.frombuffer(buf_labels, dtype=np.uint8).astype(np.int64)

    if shuffle is True:
        rand_id = np.random.randint(real_num, size=num_images)
        images = images[rand_id, :]
        labels = labels[rand_id,]
    images = images.reshape(num_images, image_size * image_size)
    return images, labels

# file path
data_path = 'D:/Code/python/mat3533/project/data/'
train_images_path = os.path.join(data_path, 'train-images-idx3-ubyte.gz')
train_labels_path = os.path.join(data_path, 'train-labels-idx1-ubyte.gz')
test_images_path = os.path.join(data_path, 't10k-images-idx3-ubyte.gz')
test_labels_path = os.path.join(data_path, 't10k-labels-idx1-ubyte.gz')

# get data
train_images, train_labels = get_mnist_data(
    train_images_path, train_labels_path, 60000)
test_images, test_labels = get_mnist_data(
```

```
    test_images_path, test_labels_path, 10000, _is=False)
print(train_images.shape, train_labels.shape)
print(test_images.shape, test_labels.shape)
```

Định nghĩa hàm `get_mnist_data` để tải dữ liệu từ tệp gzip. Hàm nhận vào các tham số:

`images_path`: đường dẫn đến tệp dữ liệu hình ảnh

`labels_path`: đường dẫn đến tệp nhãn

`num_images`: số lượng hình ảnh cần tải

`shuffle`: cho phép trộn dữ liệu, mặc định là `False`

`_is`: chỉ ra đây là dữ liệu huấn luyện hay kiểm tra, mặc định là `True`

`image_size`: kích thước của hình ảnh, mặc định là 28px x 28px

Hàm trả về một mảng chứa hình ảnh và một mảng chứ nhãn tương ứng.

Kết quả khi chạy đoạn code trên là:

```
(60000, 784) (60000,)
(10000, 784) (10000,)
```

Từ kết quả nhận được, ta thấy rằng:

`train_images` là một mảng chứa 60000 hình ảnh huấn luyện, `test_images` là một mảng chứa 10000 hình ảnh kiểm tra, mỗi hình ảnh được biểu diễn dưới dạng vector có 784 phần tử

`train_labels` là một mảng chứa 60000 nhãn tương ứng với 60000 hình ảnh huấn luyện, `test_labels` là một mảng chứa 10000 nhãn tương ứng với 10000 hình ảnh kiểm tra, có giá trị (0, 1, 2.., 9)

5.1.2 Tiền xử lý dữ liệu

Sau khi lấy được dữ liệu, ta tiến hành chuẩn hóa dữ liệu bằng việc sử dụng lớp `StandardScaler` trong thư viện `Scikit-learn`:

```
    scaler = StandardScaler()
    train_x = scaler.fit_transform(train_images)
    test_x = scaler.fit_transform(test_images)
```

Tiếp đến, ta thực hiện giảm số chiều dữ liệu bằng PCA:

Giảm số chiều dữ liệu xuống 2 chiều để chuẩn bị cho việc trực quan các phân lớp dữ liệu:

```
    pca = PCA(n_components=2)
    train_pca_02 = pca.fit_transform(train_x)
    principal_df = pd.DataFrame(data = train_pca_02, columns = ['pc01', 'pc02'])
```

Giảm số chiều dữ liệu xuống 100 chiều để chuẩn bị cho việc huấn luyện các mô hình học máy:

```
pca = PCA(n_components=100)
pca.fit(train_images)
X_train = pca.transform(train_images)
X_test = pca.transform(test_images)
```

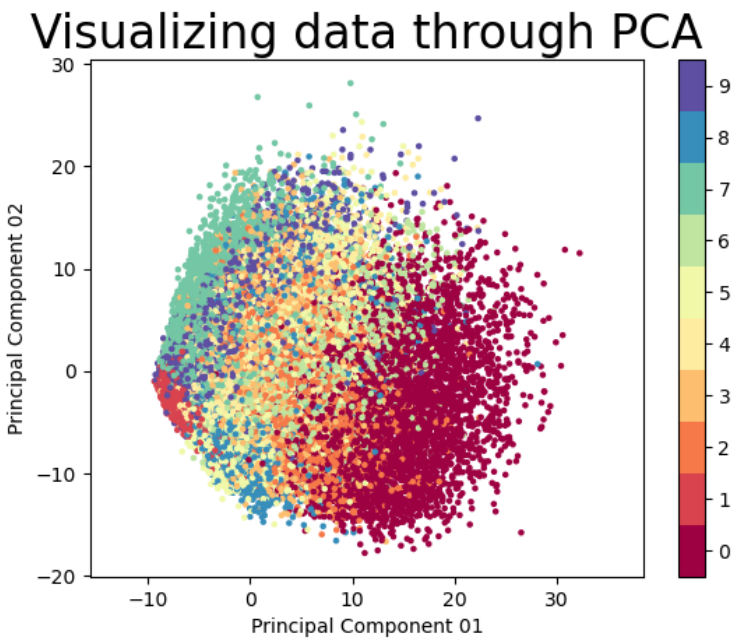
5.2 Trực quan các phân lớp dữ liệu

Ta sử dụng thư viện `matplotlib` để trực quan hóa dữ liệu ở dạng 2D sau khi đã giảm số chiều bằng PCA:

```
import matplotlib.pyplot as plt

plt.scatter(train_pca_02[:, 0], train_pca_02[:, 1], s= 5, c=train_labels,
    cmap='Spectral')
plt.gca().set_aspect('equal', 'datalim')
plt.colorbar(boundaries=np.arange(11)-0.5).set_ticks(np.arange(10))
plt.title('Visualizing data through PCA', fontsize=24)
plt.xlabel('Principal Component 01')
plt.ylabel('Principal Component 02')
```

Sau khi chạy đoạn code trên, ta thu được biểu đồ:



Hình 5.2 Trực quan các phân lớp dữ liệu ở dạng 2D

Mỗi điểm trên biểu đồ đại diện cho một hình ảnh từ tập dữ liệu, được tô màu dựa theo nhãn tương ứng.

5.3 Nhận dạng hình ảnh chữ số viết tay bằng mô hình Gausian Naive Bayes

5.3.1 Huấn luyện mô hình và dự đoán nhãn tập kiểm tra

Ta sử dụng thư viện `scikit-learn` để xây dựng, huấn luyện, dự đoán hình ảnh chữ số viết tay bằng mô hình Gaussian Naive Bayes. Quá trình này được thực hiện như sau:

- Import thư viện cần thiết:

```
from sklearn.naive_bayes import GaussianNB
```

- Khởi tạo mô hình:

```
model = GaussianNB()
```

- Huấn luyên mô hình:

```
model.fit(X_train, train_labels)
```

- Dự đoán nhãn cho tập kiểm tra:

```
pred_labels = model.predict(X_test)
```

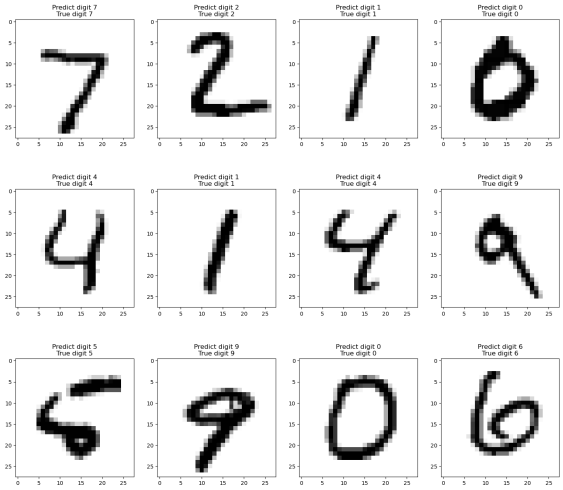
5.3.2 Trực quan hóa một số dự đoán của mô hình

Ta sử dụng đoạn code sau để hiển thị trực quan 12 dự đoán của mô hình:

```
import matplotlib.pyplot as plt

fig, axis = plt.subplots(3, 4, figsize=(18, 16))
for i, ax in enumerate(axis.flat):
    ax.imshow(test_images[i].reshape(28, 28), cmap='binary')
    ax.set(title = "Predict digit {0}\nTrue digit {1}".format(pred_labels[i],
        test_labels[i]))
```

Kết quả hiển thị:



Hình 5.3 Trực quan một số dự đoán của mô hình Naive Bayes

5.4 Nhận dạng hình ảnh chữ số viết tay bằng mô hình ANN

5.4.1 Huấn luyện mô hình và dự đoán nhãn tập kiểm tra

Ta sử dụng thư viện TensorFlow và Keras để xây dựng, huấn luyện, dự đoán hình ảnh chữ số viết tay bằng mô hình ANN. Quá trình này được thực hiện như sau

- Import các thư viện:

```
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten
```

- Khởi tạo mô hình `Sequential`:

```
model = Sequential()
```

- Thêm các lớp `layer` vào mô hình:

```
model.add(Flatten(input_shape = (100,)))
model.add(Dense(128,activation = 'relu'))
model.add(Dense(64,activation = 'relu'))
model.add(Dense(32,activation = 'relu'))
model.add(Dense(10,activation = 'softmax'))
```

- Compile mô hình

```
model.compile(loss = 'sparse_categorical_crossentropy',
      optimizer = 'Adam',
      metrics = ['accuracy'])
```

- Huấn luyện mô hình:

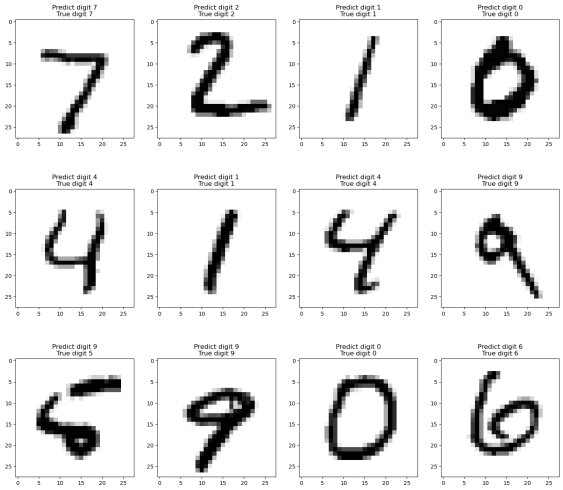
```
model.fit(X_train, train_labels, epochs= 10 , validation_split = .2)
```

- Dự đoán nhãn cho tập kiểm tra:

```
y_pred = model.predict(X_test)
y_pred = y_pred.argmax(axis = 1)
```

5.4.2 Trực quan hóa một số dự đoán của mô hình

Hiển thị trực quan 12 dự đoán của mô hình:



Hình 5.4 Trực quan một số dự đoán của mô hình ANN

5.5 So sánh độ chính xác hai mô hình Naive Bayes và ANN

5.5.1 Kết quả các chỉ số của hai mô hình

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

```
from sklearn.metrics import accuracy_score, confusion_matrix,
    precision_score, recall_score

train_time = end_time - start_time
print("Train time: ", train_time)

accuracy = accuracy_score(test_labels, pred_labels)
print("Accuracy:", accuracy)

confusion_matrix = confusion_matrix(test_labels, pred_labels)
print("Confusion Matrix:")
print(confusion_matrix)

precision = precision_score(test_labels, pred_labels, average='macro')
print("Precision:", precision)

recall = recall_score(test_labels, pred_labels, average='macro')
print("Recall:", recall)
```

Các chỉ số của mô hình Gausian Naive Bayes

```
Train time: 0.05646944046020508
Accuracy: 0.8788
Confusion Matrix:
[[ 916    0   35    2    0   13   12    1    0    1]
 [   0 1070    8   10    3   11    7    3   20    3]
 [  16    0  880   57   10    4    9    9   46    1]
 [   4    0   36  873    0   32    5   19   26   15]
 [   0    0   36    1  851    6    8    4    6   70]
 [   5    1    9   57    3  766    8    4   31    8]
 [   5    2   38    2    9   34  867    0    1    0]
 [   4   19   41    5   21   14    3  861   14   46]
 [  11    0   26   27    7   28    7    8  850   10]
 [  12    1   31    7   47   20    1   25   11  854]]
Precision: 0.8801310384842731
Recall: 0.8781469513576236
```

Các chỉ số của mô hình ANN

```
Train time: 18.097848415374756
Accuracy: 0.9651
Confusion Matrix:
[[ 973    0    2    0    0    1    1    1    2    0]
 [   0 1112    2    3    1    6    1    2    8    0]
 [   4    1  997    8    5    3    1    5    8    0]
 [   0    0    8  983    0    6    0    5    6    2]
 [   1    0    8    0  951    2    1    2    2   15]
 [   8    0    0    9    3  854    6    1    9    2]
 [   5    2    9    0    3   18  913    0    8    0]
 [   2    5   23    2    0    4    0  982    5    5]
 [   4    0    7    8    0    2    2    5  946    0]
 [   4    3    0    9    8    8    0   11   26  940]]
Precision: 0.9650317598381282
Recall: 0.9648924384510256
```

5.5.2 So sánh hai mô hình

Yếu tố	**Mô hình Naive Bayes**	**Mô hình ANN**
Accuracy	87.88%	96.51%
Precision	88.01%	96.50%
Recall	87.81%	96.49%
Training time	0.056 giây	18.098 giây

Bảng 5.5 Bảng so sánh các chỉ số hai mô hình Naive Bayes và ANN

- Về `accuracy`, mô hình ANN có độ chính xác cao hơn đáng kể so với mô hình Naive Bayes
- Về `precision` và `recall`, mô hình ANN cao hơn nhiều so với mô hình Naive Bayes cho thấy mô hình ANN có khả năng dự đoán chính xác và bao quát tốt hơn trên các lớp
- Về `confusion matrix`, Các số liệu trên đường chéo chính của ma trận `confusion` của mô hình ANN cao và ít nhầm lẫn hơn mô hình Gaussian Naive Bayes. Điều này cho thấy, mô hình ANN có xu hướng dự đoán chính xác hơn mô hình Naive Bayes
- Về thời gian huấn luyện, mô hình Naive Bayes có thời gian huấn luyện nhanh hơn khá nhiều so với mô hình ANN

Từ các chỉ số trên ta thấy được, cả hai mô hình đều có hiệu suất làm việc khá tốt. Tuy nhiên, mô hình ANN có độ chính xác tốt hơn mô hình Naive Bayes. Mô hình Naive Bayes có thời gian huấn luyện nhanh chóng, nhưng việc "học" các mối quan hệ phức tạp giữa các đặc trưng của dữ liệu chưa thực sự tốt. Điều đó dẫn đến việc nhận dạng chữ số có nhiều đặc trưng giống nhau bằng mô hình này chưa đáng tin cậy. Bên cạnh đó, mô hình ANN phức tạp hơn và tốn nhiều thời gian huấn luyện hơn, nhưng khả năng "học" mối quan hệ phức tạp giữa các đặc trưng của dữ liệu và việc dự đoán trên tập kiểm tra cao hơn đáng kể so với mô hình Naive Bayes. Vì vậy mô hình ANN có độ chính xác cao hơn và đáng tin cậy hơn trong việc nhận dạng chữ số viết tay so với mô hình Gausian Naive Bayes.

KẾT LUẬN

Nhìn chung, mô hình Gaussian Naive Bayes là một mô hình đơn giản, huấn luyện nhanh chóng tuy nhiên việc "học" các mối quan hệ phức tạp giữa các đặc trưng của dữ liệu chưa thực sự tốt. Bên cạnh đó, mô hình ANN phức tạp hơn và tốn nhiều thời gian huấn luyện hơn, nhưng khả năng "học" mối quan hệ phức tạp giữa các đặc trưng của dữ liệu và việc dự đoán trên tập kiểm tra tốt hơn so với mô hình Naive Bayes. Từ đây có thể nói mô hình ANN có độ chính xác cao hơn và đáng tin cậy hơn trong việc nhận dạng chữ số viết tay so với mô hình Naive Bayes. Tuy nhiên, ta vẫn cần xem xét đến những hướng phát triển mới cho mô hình nhằm nâng cao hiệu quả, cải thiện hiệu suất bằng cách xem xét thử nghiệm trên các biến thể của Naive Bayes, áp dụng các kỹ thuật smoothing và tối ưu hóa các siêu tham số để cải thiện hiệu suất của mô hình. Hay sử dụng các kiến trúc mạng phức tạp hơn như các mạng nơ-ron sâu (deep neural networks) hoặc các mô hình tái sinh (autoencoders), Tăng cường tập dữ liệu huấn luyện và sử dụng các kỹ thuật tăng cường dữ liệu như xoay, phóng đại, và cắt ghép để mở rộng tập dữ liệu huấn luyện.

TÀI LIỆU THAM KHẢO

1. Géron, Aurélien. "Hands-On Machine Learning with Scikit-Learn, Keras, and Tensor-Flow." O'Reilly Media, 2019.
2. Goodfellow, Ian, Yoshua Bengio, và Aaron Courville. "Deep Learning." MIT Press, 2016.
3. Bishop, Christopher M. "Pattern Recognition and Machine Learning." Springer, 2006.
4. TensorFlow Documentation. Truy cập từ: https://www.tensorflow.org/
5.Scikit-learn Documentation. Truy cập từ: https://scikit-learn.org/
6. Keras Documentation. Truy cập từ: https://keras.io/
7. Hastie, Trevor, Robert Tibshirani, và Jerome Friedman. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction." Springer, 2009.
8. Murphy, Kevin P. "Machine Learning: A Probabilistic Perspective." MIT Press, 2012.