ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA TOÁN TIN

HỒI QUY SOFTMAX VÀ ỨNG DỤNG TRONG BÀI TOÁN PHÂN LOẠI CHỮ CÁI VIẾT TAY

Chuyên sâu: Toán ứng dụng

Giảng viên hướng dẫn: TS. Phạm Thị Hoài
Sinh viên thực hiện: Đỗ Thế Anh
MSSV: 20216792
Lớp: Toán tin 02 - K66

Hà Nội, 2024

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục tiêu và nội dung của đồ án

(a) Mục tiêu: Đề tài tìm hiểu về mô hình hồi quy Softmax và ứng dụng trong bài toán phân loại chữ cái viết tay. Sử dụng thuật toán hướng giảm gradient với cỡ bước hằng và với thuật toán quay lui.

(b) Nội dung: Trình bày tổng quan cơ sở toán học; mô hình hồi quy Softmax; cách xây dựng hàm mất mát theo hàm entropy chéo và cách tối ưu hàm mất mát bằng thuật toán hướng giảm gradient với cỡ bước hằng và thuật toán quay lui; lập trình mô hình phân loại chữ cái viết tay.

2. Kết quả đạt được

(a) Sinh viên hiểu mô hình hồi quy Softmax, và thuật toán hướng giảm gradient.

(b) Sinh viên biết cách áp dụng thuật toán hướng giảm gradient với cỡ bước hằng và với thuật toán quay lui để tối ưu hàm mất mát thu được từ mô hình hồi quy Softmax.

(c) Áp dụng cho bài toán phân loại chữ cái viết tay.

3. Ý thức làm việc của sinh viên:

(a)

(b)

(c)

Hà Nội, ngày 20 tháng 6 năm 2024

Giảng viên hướng dẫn

TS. Phạm Thị Hoài

Mục lục

Chương 1 Kiến thức chuẩn bị **4**

1.1 Một số khái niệm cơ bản của giải tích lồi 4

1.1.1 Đạo hàm riêng 4

1.1.2 Gradient 5

1.1.3 Đạo hàm theo hướng 5

1.1.4 Hàm lồi 6

1.2 Hướng giảm 8

1.2.1 Phương pháp hướng giảm Gradient 9

1.3 Các tiêu chí đánh giá mô hình phân lớp 10

1.3.1 Ma trận nhầm lẫn (Confusion matrix) 10

1.3.2 Accuracy, Precision, Recall và F1-score 12

Chương 2 Hàm Softmax **14**

2.1 Cách hoạt động của hồi quy Softmax 14

2.2 Hàm Softmax 15

2.2.1 Tổng quát và véc tơ hóa 16

2.2.2 Phiên bản ổn định hơn của hàm Softmax 19

2.2.3 Hồi quy Logistic là trường hợp đặc biệt của hồi quy Softmax 20

Chương 3 Hàm Entropy chéo và tối ưu hàm mất mát **22**

3.1 Hàm entropy chéo 22

3.1.1 Công thức hàm entropy chéo 22

3.1.2 Mối liên hệ giữa hàm entropy chéo, hàm entropy và độ lệch KL 23

3.2 Xây dựng hàm mất mất và tối ưu hàm mất mát cho mô hình hồi quy Softmax 26
3.2.1 Mã hóa one-hot 26
3.2.2 Xây dựng hàm mất mát 28
3.2.3 Tối ưu hàm mất mát 30
Chương 4 Ứng dụng hồi quy Softmax vào bài toán nhận dạng chữ cái viết tay **32**
4.1 Giới thiệu bộ dữ liệu *letter* 32
4.1.1 Tổng quan bộ dữ liệu *letter* 32
4.1.2 Thống kê bộ dữ liệu 33
4.2 Huấn luyện mô hình bằng thuật toán hướng giảm gradient 34
4.2.1 Huấn luyện mô hình với cỡ bước hằng 34
4.2.2 Huấn luyện với hướng giảm gradient và thuật toán quay lui 40
4.2.3 So sánh và đánh giá 45
Kết luận **48**
Tài liệu tham khảo **49**

Lời mở đầu

Trong kỷ nguyên số hóa hiện nay, việc phân loại và dự đoán các dữ liệu phức tạp trở thành một trong những thách thức lớn nhất đối với các nhà nghiên cứu và kỹ sư trong lĩnh vực học máy và trí tuệ nhân tạo. Hồi quy Softmax là một trong những thuật toán học máy cơ bản nhưng mạnh mẽ, đóng vai trò then chốt trong việc giải quyết các vấn đề phân loại đa lớp. Đây là một biến thể mở rộng của hồi quy Logistic, cho phép phân loại dữ liệu thành nhiều hơn hai lớp.

Hồi quy Softmax không chỉ là một công cụ phân loại hiệu quả mà còn là nền tảng cho nhiều mô hình học sâu phức tạp. Với khả năng tối ưu hóa và tính toán xác suất cho từng lớp trong tập dữ liệu, thuật toán này được ứng dụng rộng rãi trong nhiều lĩnh vực, từ nhận diện hình ảnh, phân tích văn bản đến dự đoán hành vi người dùng.

Đồ án tập trung vào việc tìm hiểu về mô hình tối ưu của bài toán hồi quy Softmax bao gồm cơ sở lý thuyết đến và ứng dụng thực tiễn, cấu trúc toán học, cách hoạt động của thuật toán hồi quy Softmax, cùng với phương pháp tối ưu hóa hàm mất mát để cho ra bộ tham số có độ chính xác với mô hình hồi quy Softmax.

Ngoài ra, đồ án còn trình bày kết quả thực nghiệm khi áp dụng hồi quy Softmax trên tập dữ liệu *letter*, và sẽ phân tích hiệu suất của mô hình và đánh giá độ hiệu quả của thuật toán với bộ dữ liệu này.

Đặc biệt, em xin gửi lời cảm ơn chân thành và sâu sắc tới TS. Phạm Thị Hoài, người đã trực tiếp hướng dẫn em hoàn thành đồ án này. Cô đã tận tâm chỉ dẫn, luôn theo dõi sát sao và đưa ra những lời khuyên vô cùng hữu ích, giúp

em vượt qua những khó khăn và giải quyết các vấn đề gặp phải trong quá trình làm báo cáo. Nhờ đó, em đã có thể hoàn thành đồ án một cách tốt nhất.

Mặc dù đã nỗ lực hết sức, nhưng do kiến thức chuyên môn của em còn hạn chế và kinh nghiệm thực tiễn còn ít ỏi, nên nội dung của đồ án không tránh khỏi những sai sót. Em rất mong nhận được sự góp ý và chỉ bảo thêm của quý thầy cô để em có thể cải thiện và hoàn thiện đồ án này hơn nữa trong tương lai. Những góp ý quý báu của quý thầy cô sẽ là nguồn động viên lớn giúp em tiếp tục học hỏi và nâng cao kiến thức, kỹ năng của mình.

Ký hiệu	Ý nghĩa
$\mathbf{x}$	Véc tơ đặc trưng
y	Nhãn thực tế
$\hat{\mathbf{y}}$	Véc tơ xác suất dự đoán
$\hat{y}$	Nhãn dự đoán
$\mathbf{X}$	Ma trận điểm dữ liệu
θ	Véc tơ trọng số
Θ	Ma trận trọng số
$\Theta^\top$	Chuyển vị của ma trận trọng số
$\mathbf{z}$	Véc tơ đầu vào
$\sigma(z)$	Hàm Softmax
$P(y = j\|\mathbf{x}, \theta)$	Xác suất dự đoán
N	Số lượng mẫu
n	Số lượng đặc trưng
K	Số lượng lớp
$J(\theta)$	Hàm mất mát
$\nabla J(\theta)$	Gradient của hàm mất mát
$\mathbb{R}$	Tập số thực
$\mathbb{R}^n$	Không gian Euclid n chiều
$\sum$	Tổng sigma
$\|.\|$	Chuẩn
$\in$	Thuộc
$\subseteq$	Tập con

Bảng 1: **Bảng chú giải ký hiệu toán học**

Chương 1

Kiến thức chuẩn bị

Chương này, trình bày các khái niệm và kết quả liên quan đế mô hình hồi quy Softmax như đạo hàm riêng, gradient, hàm lồi, phương pháp tối ưu trong quá trình huấn luyện mô hình đó là phương pháp hướng giảm với thuật toán quay lui (quy tắc Armijo). Cuối cùng là các chỉ số đánh giá mô hình phân lớp đồng thời giải thích ý nghĩa của chúng như Precision, Recall, F1-score và ma trận nhầm lẫn.
Nội dung tham khảo chủ yếu trong [2].

1.1 Một số khái niệm cơ bản của giải tích lồi

1.1.1 Đạo hàm riêng

Định nghĩa 1.1 (Đạo hàm riêng) [2]-tr.8
Cho hàm số f xác định trên tập mở $X \subseteq \mathbb{R}^n$ và $x^0 = (x_1^0, \cdots, x_n^0)^T$ là một điểm thuộc X. Khi đó với mối số $h \in \mathbb{R}$ đủ nhỏ, điểm:

$$(x_1^0, \cdots, x_{i-1}^0, x_i^0 + h, x_{i+1}^0, \cdots, x_n^0)^T$$

cũng nằm trong X. Giới hạn

$$\lim_{h \to 0} \frac{f(x_1^0, \ldots, x_{1-1}^0, x_1^0 + h, x_{i+1}^0, \ldots, x_n^0) - f(x_1^0, \ldots, x_{1-1}^0, x_i^0, x_{i+1}^0, \ldots, x_n^0)}{h},$$

nếu tồn tại, được gọi là *đạo hàm riêng của f theo biến x_i tại điểm x^0*, ký hiệu

là

$$\frac{\partial f(x^0)}{\partial x_i} \text{ hay } \quad f'_{x_i}(x^0).$$

1.1.2 Gradient

Định nghĩa 1.2 (Gradient) [2]-tr.9

Cho hàm $f(x_1, x_2, \ldots, x_n)$ xác định trên tập mở $X \subseteq \mathbb{R}^n$. Giả sử rằng tại x^0, các đạo hàm riêng hàm f theo mọi biến tồn tại. Khi đó, véc tơ

$$\left(\frac{\partial f(x^0)}{\partial x_1}, \frac{\partial f(x^0)}{\partial x_2}, \cdots, \frac{\partial f(x^0)}{\partial x_n}\right)^T$$

được gọi là *gradient* của f tại x^0 và ký hiệu là $\nabla f(x^0)$.

1.1.3 Đạo hàm theo hướng

Định nghĩa 1.3 (Đạo hàm theo hướng) [2]-tr.14

Cho hàm f xác định trên $\mathbb{R}^n$ và một vector $d \in \mathbb{R}^n \setminus \{0\}$. Giới hạn

$$\lim_{t \to 0^+} \frac{f(x^0 + td) - f(x^0)}{t}$$

nếu tồn tại tồn tại (hữu hạn hoặc vô cùng), được gọi là đạo hàm theo hướng của hàm f tại điểm $x^0 \in \mathbb{R}^n$ và được ký hiệu là $f'(x^0, d)$.

Mệnh đề 1.1 *[2]-tr.14*

Cho hàm f xác định trên $\mathbb{R}^n$ và điểm $x^0 \in \mathbb{R}^n$. Nếu hàm f khả vi tại điểm x^0 thì

$$f'(x^0, d) = \langle \nabla f(x^0), d \rangle \forall d \in \mathbb{R}^n \setminus \{0\}.$$

Chứng minh. Vì f khả vi tại x^0 nên $\forall d \in \mathbb{R}^n \setminus \{0\}$ ta có

$$\lim_{t \to 0^+} \frac{f(x^0 + td) - f(x^0) - \frac{\langle \nabla f(x^0), d \rangle}{||d||}}{t||d||} = 0,$$

do đó

$$\frac{f(x^0 + td) - f(x^0) - \langle \nabla f(x^0), d \rangle}{||d||} = 0.$$

Và ta nhận được điều phải chứng minh. □

Nhận xét 1.1 [2]-tr.15

Đặt $\varphi(t) := f(x^0 + td)$. khi đó, theo định nghĩa ta có

$$\varphi'(0) = \frac{d\varphi(t)}{dt}|_{t=0} = \lim_{t\to 0^+} \frac{\varphi(t) - \varphi(0)}{t} = \lim_{t\to 0^+} \frac{f(x^0 + td) - f(x^0)}{t} = f'(x^0, d).$$

Như vậy đạo hàm theo hướng của hàm f tại x^0 phản ánh tốc độ biến thiên của hàm f tại x^0 theo hướng đó. Hơn nữa, theo bất đẳng thức Cauchy-Bunijakowski-Schwarz, trong tất cả các hướng $d \in \mathbb{R}^n$ có $\|d\| = 1$, ta có

$$|\langle \nabla f(x^0), d\rangle| \le \|\nabla f(x^0)\|d\| = \|\nabla f(x^0)\|$$
$$\Rightarrow -\|\nabla f(x^0)\| \le \langle \nabla f(x^0), d\rangle \le \|\nabla f(x^0)\|.$$

Do đó, đạo hàm theo hướng của hàm f tại x^0 đã cho là lớn nhất khi hướng $d = \frac{\nabla f(x^0)}{\|\nabla f(x^0)\|}$ và nhỏ nhất khi $d = -\frac{\nabla f(x^0)}{\|\nabla f(x^0)\|}$ hay giá trị hàm tăng nhất theo hướng gradient và giảm nhanh nhất theo ngược hướng với gradient *(Đây sẽ là cơ sở xây dựng phương pháp hướng giảm gradient ở phần sau)*

1.1.4 Hàm lồi

Định nghĩa 1.4 [2]-tr.29

Hàm f được gọi là *hàm lồi (convex function)* xác định trên tập lồi $X \subseteq \mathbb{R}^n$ nếu

$$f(\lambda x^1 + (1-\lambda)x^2) \le \lambda f(x^1) + (1-\lambda)f(x^2)$$

với bất kỳ $x^1, x^2 \in X$ và số thực $\lambda \in [0, 1]$. Ta gọi f là *hàm lồi chặt (strictly convex functon)* trên tập lồi X nếu

$$f(\lambda x^1 + (1-\lambda)x^2) < \lambda f(x^1) + (1-\lambda)f(x^2)$$

Hàm f được gọi là *hàm lõm (concave function) (t.ư, hàm lõm chặt (strictly concave function))* nếu $-f$ là hàm lồi (t.ư, hàm lồi chặt)

Các phép toán về hàm lồi [2]-tr.32

Cho hàm lồi f_1 xác định trên tập lồi $X_1 \subseteq \mathbb{R}^n$, hàm lồi f_2 xác định trên tập lồi $X_2 \subseteq \mathbb{R}^n$ và số thực $\lambda > 0$. Các phép toán λf_1, $f_1 + f_2$, $\max\{f_1, f_2\}$ được định nghĩa như sau:

$$(\lambda f_1)(x) := \lambda f_1(x),\ x \in X_1;$$
$$(f_1 + f_2)(x) := f_1(x) + f_2(x),\ \ x \in X_1 \cap X_2;$$
$$max\{f_1, f_2\}(x) := \max\{f_1(x), f_2(x)\},\ \ x \in X_1 \cap X_2.$$

Tiêu chuẩn nhận biết hàm lồi khả vi [2]-tr.34

Định lý 1.1 *cho f là hàm khả vi trên tập lồi mở $X \subseteq \mathbb{R}^n$. Khi đó:*

i. Hàm f là hàm lồi trên X khi và chỉ khi

$$f(y) - f(x) \geq \langle \nabla f(x), y - x \rangle \quad \forall x, y \in X;$$

ii. Hàm f là hàm lõm trên X khi và chỉ khi

$$f(y) - f(x) \leq \langle \nabla f(x), y - x \rangle \quad \forall x, y \in X.$$

Định lý 1.2 *Cho f là hàm khả vi hai lần trên tập lồi mở $X \subseteq \mathbb{R}^n$. Khi đó,*

i. Hàm f là hàm lồi trên X khi và chỉ khi ma trận Hesse $\nabla^2 f(x)$ là nửa xác định dương trên X, tức với mỗi $x \in X$,

$$y^T \nabla^2 f(x) y \geq 0 \quad \forall y \in \mathbb{R}^n.$$

Hàm f là hàm lồi chặt trên X nếu $\nabla^2 f(x)$ xác định dương trên X, tức với mỗi $x \in X$,

$$y^T \nabla^2 f(x) y > 0 \quad \forall y \in \mathbb{R}^n \setminus \{0\};$$

ii. Hàm f là hàm lõm trên X khi và chỉ khi ma trận Hesse $\nabla^2 f(x)$ là nửa xác định âm trên X, tức với mỗi $x \in X$,

$$y^T \nabla^2 f(x) y \leq 0 \quad \forall y \in \mathbb{R}^n.$$

Hàm f là hàm lõm chặt trên X nếu $\nabla^2 f(x)$ xác định âm trên X, tức là với mỗi $x \in X$,

$$y^T \nabla^2 f(x) y < 0 \quad \forall y \in \mathbb{R}^n \setminus \{0\}.$$

Chứng minh. Xem [28], trang 90-91

Tính lồi lõm của một số hàm số cơ bản

Với hàm một biến f xác định trên tập lồi $X \subseteq \mathbb{R}$ ta có: *f là hàm lồi trên X khi và chỉ khi* $f''(x) \geq 0 \ \forall x \in X$.

i. Hàm $f(x) = e^x$ *là lồi trên* $\mathbb{R}$.

ii. Hàm $g(x) = -ln(x)$ *làm lồi trên* $(0, +\infty)$.

iii. Hàm $h(x) = -(gx) = ln(x)$ *làm lõm trên* $(0, +\infty)$.

Chứng minh.

i: Xét đạo hàm bậc 1 của $f(x) = e^x$

$\Rightarrow f'(x) = e^x$, tiếp tục xét đạo hàm bậc 2

$\Rightarrow f''(x) = e^x$. Do e^x luôn dương với mọi $x \in \mathbb{R}$,

tức là: $f''(x) = e^x > 0 \quad \forall x \in \mathbb{R}$. $\Rightarrow f(x) = e^x$ là hàm lồi trên $\mathbb{R}$.

ii : Xét đạo hàm bậc 1 của $g(x) = -\ln(x)$

$\Rightarrow g'(x) = -\frac{1}{x}$, tiếp tục xét đạo hàm bậc 2,

$\Rightarrow g''(x) = \frac{1}{x^2}$.

Do $\frac{1}{x^2}$ luôn dương với mọi $x \in (0, +\infty)$,

tức là: $g''(x) = \frac{1}{x^2} > 0 \quad \forall x \in (0, +\infty)$. $\Rightarrow g(x) = -ln(x)$ là hàm lồi trên $(0, +\infty)$.

iii : Vì $-ln(x)$ là hàm lồi trên $(0, +\infty) \Rightarrow -(-ln(x)) = ln(x)$ là hàm lõm trên $(0, +\infty)$. □

1.2 Hướng giảm

Cơ sở lý thuyết

Định nghĩa 1.5 [2]-tr.226

Cho $x^0 \in \mathbb{R}^n$ và một véc tơ $d \in \mathbb{R}^n$, véc tơ d được gọi là *hướng giảm (descent direction)* của hàm f nếu tồn tại $\varepsilon > 0$ ($\varepsilon \in \mathbb{R}$) và số dương t thỏa mãn

$$f(x + td) < f(x), \quad (0 < t < \varepsilon).$$

Mệnh đề 1.2 *[2]-tr.226*

Cho $x^0 \in \mathbb{R}^n$, mọi véc tơ $d \in \mathbb{R}^n$ thỏa mãn

$$(\nabla f(x^0), d) < 0$$

thì d là hướng giảm của f tại x^0 với f khả vi tại x^0.

Chứng minh. Vì f khả vi tại x^0 nên theo Mệnh đề 1.1 và giả thiết của Mệnh đề 1.2, ta có

$$f'(x^0, d) = \lim_{t \to 0^+} \frac{f(x^0 + td) - f(x^0)}{t} = \langle \nabla f(x^0), d \rangle < 0.$$

Do đó, $f(x^0 + td) - f(x^0) < 0$ với t đủ nhỏ. Mệnh đề đã được chứng minh. □

Vấn đề: Xét giá trị của hàm f tại x^0 (giả thiết $\nabla f(x^0) \neq 0$) thì giá trị của hàm f sẽ giảm nhanh nhất khi đi theo hướng nào, hay chúng ta phải tìm một hướng giảm d sao cho $\langle \nabla f(x^0), d \rangle < 0$ bé nhất. Để so sánh được được với tất cả các hướng giảm d có thể có, ta phải chuẩn hóa các véc tơ d sao cho $\|d\| = 1$.

1.2.1 Phương pháp hướng giảm Gradient

Thuật toán hướng giảm tổng quát

Thuật toán 1.1

1. Khởi tạo tùy ý $x^0 \in \mathbb{R}^n$.
2. Gán $k := 0$.
3. **If** $\nabla f(x^k) \approx 0$ hoặc $\|f(x^k) - f(x^{k-1}\| < \varepsilon$ **Then** trả về giá trị $x^* = x^k$, dừng thuật toán.
4. **Else** tính $x^{k+1} := x^k + t_k d^k$.
5. Gán $k := k + 1$, quay về bước 3.

Trong đó, $d^k \in \mathbb{R}^n$ là *hướng giảm*, t_k là *cỡ bước (độ dài bước)*. Cách chọn cỡ bước t_k và hướng giảm d^k sẽ cho ta thuật toán cụ thể tương ứng với t_k, d^k ta chọn.

Thuật toán hướng giảm gradient với thủ tục quay lui *(quy tắc Armijo)*

Thuật toán 1.2

1. Chọn $m_1 \in (0,1)$, $\alpha \in (0,1)$ và $\varepsilon > 0$ đủ nhỏ, $x^0 \in \mathbb{R}^n$ tùy ý ($\nabla f(x^0) \neq 0$), gán $k := 0$.

2. Gán $t_k := 1$.

3. Tính $x^{k+1} := x^k - t_k \nabla f(x^k)$ và $f(x^{k+1})$.

4. **If** $f(x^{k+1}) - f(x^k) \leq m_1 t_k \langle \nabla f(x^k), -\nabla f(x^k) \rangle = -m_1 t_k \|\nabla(x^k)\|^2$ **Then** Chuyển sang bước 5. **Else** Quay về bước 3.

5. Tính $\nabla f(x^{x+1})$.

6. **If** $\|\nabla f(x^{k+1})\| < \varepsilon$ **Then** trả về $x^* \approx x^{k+1}$, dừng thuật toán.
   Else $k := k + 1$, quay về bước 2.

Định lý 1.3 *(Định lý hội tụ của thuật toán Gradient với thủ tục quay lui) [2]-tr.236*
Giả sử hàm $f(x)$ bị chặn dưới, gradient $\nabla f(x)$ thỏa mãn điều kiện Lipschitz, tức tồn tại $L > 0$ sao cho

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \ \ \forall x, y \in \mathbb{R}^n.$$

Khi đó, với bất kỳ điểm xuất phát x^0, dãy x^k được chọn như trong thuật toán 1.2 có tính chất $\|\nabla f(x^k)\| \to 0$ khi $k \to \infty$.

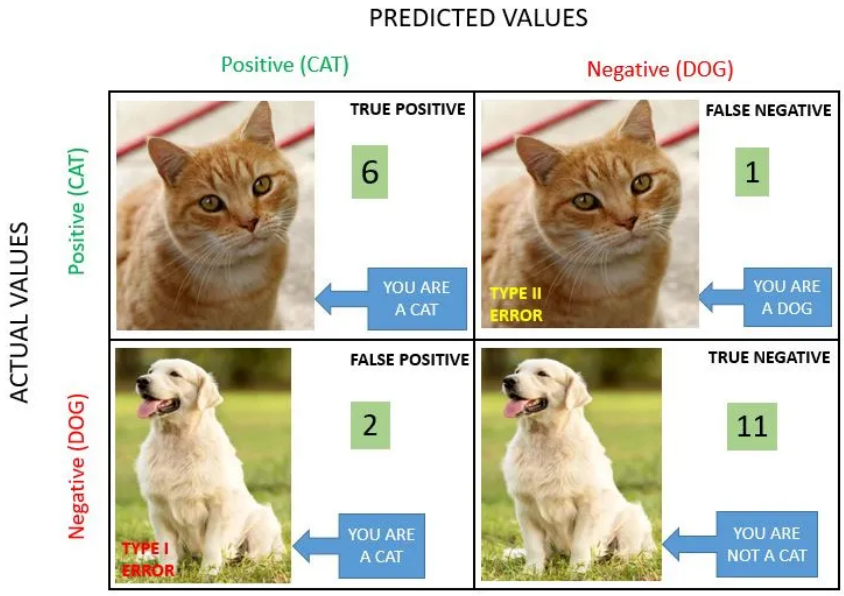
1.3 Các tiêu chí đánh giá mô hình phân lớp

1.3.1 Ma trận nhầm lẫn (Confusion matrix)

Ma trận nhầm lẫn (Confusion Matrix) là một bảng dữ liệu được sử dụng để đánh giá hiệu suất của một mô hình phân loại. Nó cho phép chúng ta xem xét cụ thể các kết quả dự đoán của mô hình so với thực tế.

	Predicted Positive	Predicted Negative
Actual Positive	**True Positive (TP)**	**False Negative (FN)**
Actual Negative	**False Positive (FP)**	**True Negative (TN)**

Bảng 1.1: Ma trận nhầm lẫn



Hình 1.1: Ví dụ về ma trận nhầm lẫn [3]

- **True Positive (TP) - Đúng dương tính:** Dự đoán đúng và nó thực sự đúng (VD: Dự đoán ảnh là con mèo, và thực sử ảnh là hình con mèo)
- **False Negative (FN) - Sai âm tính:** Hay con gọi là sai lầm loại II. Dự đoán là sai, nhưng thực tế là đúng (VD: Dự đoán ảnh không phải là con chó không phải mèo, nhưng thực tế ảnh là con mèo).
- **False Positive (FP) - Sai dương tính:** Hay còn gọi là sai lầm loại I. Dự đoán là đúng, nhưng thực tế lại sai (VD: Dự đoán ảnh là con mèo, nhưng thực tế con vật trong ảnh là con chó).
- **True Negative (TN) - Đúng âm tính:** Dự đoán không đúng và thực sự nó không đúng (VD: Dự đoán con vật trong ảnh không phải con mèo, và thực sự nó không phải con mèo mà là con chó).

1.3.2 Accuracy, Precision, Recall và F1-score

Precision

Precision là tỷ lệ của các dự đoán dương tính đúng (True Positives - TP) trên tổng số các dự đoán dương tính (bao gồm cả các dự đoán dương tính đúng và sai, tức là TP và False Positives - FP).
Precision cao nghĩa là trong số các dự đoán dương tính của mô hình, có nhiều dự đoán đúng.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}.$$

Recall

Recall là tỷ lệ của các dự đoán dương tính đúng (TP) trên tổng số các trường hợp thực sự dương tính (bao gồm cả các trường hợp dương tính mà mô hình không dự đoán được, tức là TP và False Negatives - FN).
Recall cao nghĩa là mô hình có khả năng phát hiện hầu hết các trường hợp dương tính thực sự.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}.$$

F1-score

F1-score là một chỉ số trung bình điều hòa của Precision và Recall, cung cấp một đánh giá tổng thể về hiệu suất của mô hình dự đoán.
F1-score có chỉ số cao khi cả Precision và Recall đều cao.

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Nhận xét 1.2 Đối với mô hình phân loại đa lớp, mỗi lớp sẽ có giá trị Precision, Recall và F1-score của riêng lớp đó, do đó, để đánh giá được tổng thể của mô hình ta có thể sử dụng các chỉ số trung bình có trọng số (Weighted Average) của từng chỉ số để đánh giá.

Công thức tổng quát

$$\text{Weighted Average Metric} = \frac{\sum_{i=1}^{N}(\text{Metric}_i \times \text{Support}_i)}{\sum_{i=1}^{N} \text{Support}_i}.$$

Trong đó:

- Metric_i là giá trị của chỉ số (precision, recall, F1-score) cho nhãn thứ i.
- Support_i là số lượng mẫu thuộc nhãn thứ i trong tập dữ liệu.
- N là tổng số nhãn.

Kết luận

Trong chương này, em đã trình bày được các cơ sở toán học liên quan làm nền tảng cho mô hình hồi quy Softmax như đạo hàm riêng, gradient và phương pháp tối ưu sử dụng thuật toán hướng giảm gradient để tìm bộ tham số phù hợp trong quá trình huấn luyện. Cuối cùng em đã trình bày các chỉ số đánh giá mô hình phân lớp, phục vụ cho quá trình đánh giá hiệu suất mô hình với bộ dữ liệu thực tế ở Chương 4.

Chương 2

Hàm Softmax

Ở chương này, em tập trung trình bày, đưa ra khái niệm, ý nghĩa của hàm Softmax, cách mô hình hồi quy Softmax hoạt động. Phương pháp véc tơ hóa các giá trị đầu vào và đầu ra của hàm Softmax. Bên cạnh đó, em trình bày ý nghĩa của hàm Softmax phiên bản ổn định và mối liên hệ giữa mô hình hồi quy Softmax và hồi quy Logistic.
Nội dung tham khảo chủ yếu trong [1].

2.1 Cách hoạt động của hồi quy Softmax

Giả sử, với một điểm dữ liệu $\mathbf{x} \in \mathbb{R}^n$ bao gồm n đặc trưng và có K lớp cần phân loại, thì các tham số đầu vào của thuật toán là véc tơ $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$, đại diện cho điểm dữ liệu $\mathbf{x}$, các giá trị x_i $(i = 1, ..., n)$ là các đặc trưng của dữ liệu. Sau khi đi qua hàm Softmax , giá trị trả về là một véc tơ xác suất $\hat{\mathbf{y}} \in \mathbb{R}^K$ $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_K)^T$, với $\hat{y}_j$ $(j = 1, ..., K)$ là xác suất để $\mathbf{x}$ thuộc vào lớp thứ i, và:

$$\sum_{j=1}^{K} \hat{y}_j = 1$$

Lúc này, giá trị lớn nhất của các $\hat{y}_j$ giả sử $max\{y_j\} = y_k$, thì điểm $\mathbf{x}$ sẽ được gán vào lớp k (với $1 \leq k \leq K$).

2.2 Hàm Softmax

Như ta đã biết ở phần 3.1, thì đầu ra của thuật toán hồi quy Softmax là một ma trận xác suất, để làm được điều này, thuật toán đã dùng một hàm được gọi là hàm Softmax, giúp chuyển đổi ma trận đầu vào thành một ma trận xác suất đầu ra, cụ thể sẽ được giải thích ngay sau đây.

Định nghĩa:

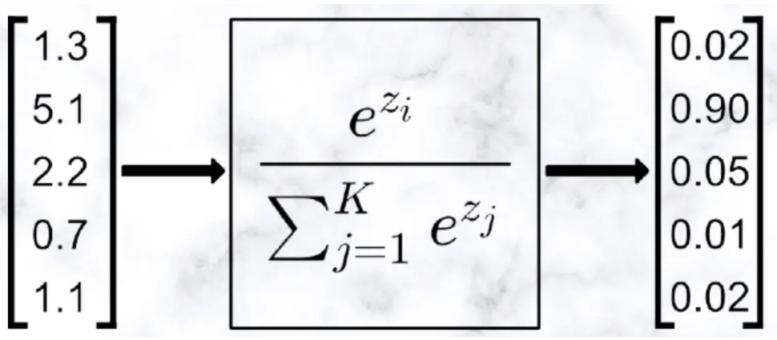
Hàm Softmax ký hiệu $\sigma : \mathbb{R}^K \to \mathbb{R}^K$ được định nghĩa như sau:

$$P(y = k|\mathbf{x}, \theta_k) = \sigma(\mathbf{z})_k = \frac{e^{\mathbf{z}_k}}{\sum_{j=1}^{K} e^{\mathbf{z}_j}}, \quad (i = 1, 2, \ldots, K). \tag{2.1}$$

Với $\mathbf{z}_k = (\theta_k)^\top \mathbf{x}$

$$P(y = k|\mathbf{x}, \theta_k) = \sigma(\mathbf{z})_k = \frac{e^{(\theta_k)^\top \mathbf{x}}}{\sum_{j=1}^{K} e^{(\theta_j)^\top \mathbf{x}}}. \tag{2.2}$$

- Véc tơ đầu vào $\mathbf{z} = (z_1, z_2, \ldots, z_K)$: $\mathbf{z} \in \mathbb{R}^K$, với z_j là thành phần thứ j của véc tơ $\mathbf{z}$.
- Véc tơ đặc trưng đại diện cho điểm dữ liệu $\mathbf{x} = (x_1, x_2, \ldots, x_n)$: $\mathbf{x} \in \mathbb{R}^n$, với x_i là thành phần thứ i của véc tơ $\mathbf{x}$.
- Véc tơ trọng số cho lớp i: $\theta_i = (\theta_{i1}, \theta_{i2}, \ldots, \theta_{in})$: $\theta_i \in \mathbb{R}^n$, với θ_{ij} là thành phần thứ j của véc tơ θ_i.
- Số lượng lớp: K



Hình 2.1: Minh họa hàm Softmax [5]

2.2.1 Tổng quát và véc tơ hóa

Giả sử bộ dữ liệu có N điểm dữ liệu, n đặc trưng và cần phân loại vào K lớp.

Biểu diễn ma trận trọng số Θ

Ma trận trọng số $\Theta \in \mathbb{R}^{n \times K}$ được biểu diễn như sau:

Ứng với mỗi lớp i có véc tơ trọng số $\theta_i = (\theta_{1i}, \theta_{2i}, \ldots, \theta_{ni})^T$

$$\Theta = \begin{pmatrix} \theta_1 & \theta_2 & \cdots & \theta_K \end{pmatrix} = \begin{pmatrix} \theta_{11} & \theta_{12} & \cdots & \theta_{1K} \\ \theta_{21} & \theta_{22} & \cdots & \theta_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{n1} & \theta_{n2} & \cdots & \theta_{nK} \end{pmatrix},$$

trong đó, mỗi cột θ_i là một véc tơ trọng số cho lớp i.

Ví dụ: nếu $n = 3$ (số lượng đặc trưng) và $K = 4$ (số lượng lớp), ma trận Θ có thể được biểu diễn như sau:

$$\Theta = \begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} & \theta_{14} \\ \theta_{21} & \theta_{22} & \theta_{23} & \theta_{24} \\ \theta_{31} & \theta_{32} & \theta_{33} & \theta_{34} \end{pmatrix},$$

trong đó:

- $\theta_1 = (\theta_{11}, \theta_{21}, \theta_{31})^T$ là véc tơ trọng số cho lớp thứ 1,
- $\theta_2 = (\theta_{12}, \theta_{22}, \theta_{32})^T$ là véc tơ trọng số cho lớp thứ 2,
- $\theta_3 = (\theta_{13}, \theta_{23}, \theta_{33})^T$ là véc tơ trọng số cho lớp thứ 3,
- $\theta_4 = (\theta_{14}, \theta_{24}, \theta_{34})^T$ là véc tơ trọng số cho lớp thứ 4.

Biểu Diễn Ma Trận Đầu Vào X

Với N điểm dữ liệu, mỗi điểm dữ liệu có n đặc trưng. Véc tơ $\mathbf{x}_i = (x_{1i}, x_{2i}, \ldots, x_{ni})^T$ là đại diện cho một điểm dữ liệu bất kỳ, khi này ta sẽ có ma trận đầu vào $\mathbf{X} \in \mathbb{R}^{n \times N}$ và được biểu diễn như sau:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nN} \end{pmatrix}.$$

Ví dụ: Giả sử $n = 3$ (số lượng đặc trưng) và $N = 4$ (số lượng điểm dữ liệu), ma trận $\mathbf{X}$ có thể được biểu diễn như sau:

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{pmatrix},$$

trong đó:

- $\mathbf{x}_1 = (x_{11}, x_{21}, x_{31})^T$ là véc tơ đặc trưng của điểm dữ liệu thứ nhất,
- $\mathbf{x}_2 = (x_{12}, x_{22}, x_{32})^T$ là véc tơ đặc trưng của điểm dữ liệu thứ hai,
- $\mathbf{x}_3 = (x_{13}, x_{23}, x_{33})^T$ là véc tơ đặc trưng của điểm dữ liệu thứ ba,
- $\mathbf{x}_4 = (x_{14}, x_{24}, x_{34})^T$ là véc tơ đặc trưng của điểm dữ liệu thứ tư.

Biểu diễn ma trận điểm số Z

Ma trận điểm số $\mathbf{Z} \in \mathbb{R}^{K \times N}$ và được tính bằng cách nhân ma trận trọng số chuyển vị $\Theta^T \in \mathbb{R}^{K \times n}$ với ma trận đầu vào $\mathbf{X} \in \mathbb{R}^{n \times N}$.

$$\mathbf{Z} = \Theta^T \mathbf{X}.$$

Ví dụ, nếu $n = 3$, $K = 4$, và $N = 4$:

$$\mathbf{Z} = \Theta^T \mathbf{X} = \begin{pmatrix} \theta_{11} & \theta_{21} & \theta_{31} \\ \theta_{12} & \theta_{22} & \theta_{32} \\ \theta_{13} & \theta_{23} & \theta_{33} \\ \theta_{14} & \theta_{24} & \theta_{34} \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{pmatrix} = \begin{pmatrix} z_{11} & z_{12} & z_{13} & z_{14} \\ z_{21} & z_{22} & z_{23} & z_{24} \\ z_{31} & z_{32} & z_{33} & z_{34} \\ z_{41} & z_{42} & z_{43} & z_{44} \end{pmatrix}.$$

Áp dụng hàm Softmax cho từng cột của ma trận $\mathbf{Z}$ để thu được ma trận xác suất $\hat{Y} \in \mathbb{R}^{K \times N}$.

Ví dụ minh họa với số liệu cụ thể

Giả sử chúng ta có 2 điểm dữ liệu với 3 đặc trưng ($N = 2, n = 3$) và 3 lớp ($K = 3$): Các điểm dữ liệu x_i được biểu diễn như sau:
$\mathbf{x}_1 = (1, 2, 3)^T$, $\mathbf{x}_2 = (4, 5, 6)^T$ từ đây, ta thu được ma trận đầu vào cho toàn bộ dữ liệu X

$$\mathbf{X} = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}.$$

Với 3 lớp, ta sẽ có 3 bộ tham số được biểu diễn như sau: $\theta_1 = (0.1, 0.4, 0.7)^T$, $\theta_2 = (0.2, 0.5, 0.8)^T$ và $\theta_3 = (0.3, 0.6, 0.9)^T$. Ma trận Θ cho bộ tham số của 3 lớp như sau:

$$\Theta = \begin{pmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \end{pmatrix}.$$

Tính ma trận điểm số $\mathbf{Z}$:

$$\mathbf{Z} = \Theta^T \mathbf{X} = \begin{pmatrix} 0.1 & 0.4 & 0.7 \\ 0.2 & 0.5 & 0.8 \\ 0.3 & 0.6 & 0.9 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} = \begin{pmatrix} 3 & 6.6 \\ 3.6 & 8.1 \\ 4.2 & 9.6 \end{pmatrix}.$$

Áp dụng hàm Softmax cho mỗi cột của $\mathbf{Z}$:

Cột 1:

$$\mathbf{z}_1 = \begin{pmatrix} 3 \\ 3.6 \\ 4.2 \end{pmatrix} \implies \sigma(\mathbf{z}_1) = \begin{pmatrix} \frac{e^3}{e^3 + e^{3.6} + e^{4.2}} \\ \frac{e^{3.6}}{e^3 + e^{3.6} + e^{4.2}} \\ \frac{e^{4.2}}{e^3 + e^{3.6} + e^{4.2}} \end{pmatrix} = \begin{pmatrix} 0.16 \\ 0.3 \\ 0.54 \end{pmatrix}.$$

Cột 2:

$$\mathbf{z}_2 = \begin{pmatrix} 6.6 \\ 8.1 \\ 9.6 \end{pmatrix} \implies \sigma(\mathbf{z}_2) = \begin{pmatrix} \frac{e^{6.6}}{e^{6.6}+e^{8.1}+e^{9.6}} \\ \frac{e^{8.1}}{e^{6.6}+e^{8.1}+e^{9.6}} \\ \frac{e^{9.6}}{e^{6.6}+e^{8.1}+e^{9.6}} \end{pmatrix} = \begin{pmatrix} 0.04 \\ 0.18 \\ 0.78 \end{pmatrix}.$$

Ma Trận Xác Suất $\hat{Y}$

$$\hat{Y} = \begin{pmatrix} 0.16 & 0.04 \\ 0.3 & 0.18 \\ 0.54 & 0.78 \end{pmatrix}.$$

2.2.2 Phiên bản ổn định hơn của hàm Softmax

Khi lập trình trên máy tính hàm Softmax thông thường có thể gặp các vấn đề liên quan đến độ chính xác số học, đặc biệt là hiện tượng tràn số (overflow) và tràn số ngược (underflow), khi các giá trị điểm số z đầu vào rất lớn hoặc rất nhỏ.

Ví dụ: - Giả sử $\mathbf{z} = (1000, 1001, 1002)$ - Tính e^{1000}, e^{1001}, và e^{1002} có thể dẫn đến các giá trị rất lớn, có thể vượt quá khả năng biểu diễn của kiểu số thực trong máy tính dẫn đến Overflow.

Để khắc phục các vấn đề trên, một kỹ thuật thông thường là trừ giá trị lớn nhất của véc tơ điểm số ra khỏi từng phần tử trong véc tơ đó trước khi áp dụng hàm mũ. Điều này không thay đổi kết quả của hàm Softmax vì việc trừ đi một hằng số từ tất cả các điểm số z không ảnh hưởng đến kết quả cuối cùng (tỷ lệ giữa các giá trị mũ không thay đổi). Ta sẽ chọn giá trị lớn nhất trong các z_i để làm hằng số trừ.

Công thức hàm Softmax ổn định

$$Softmax(z_i) = \frac{e^{z_i - \max(\mathbf{z})}}{\sum_{j=1}^{K} e^{z_j - \max(\mathbf{z})}}.$$

2.2.3 Hồi quy Logistic là trường hợp đặc biệt của hồi quy Softmax

Khi số lớp của mô hình hồi quy Softmax là 2 ($K = 2$). Ta sẽ có hai lớp là 0 và 1.

Xác suất dự đoán lớp 1:

$$P(y = 1|\mathbf{x}, \boldsymbol{\theta_1}) = \frac{e^{\boldsymbol{\theta}_1^T \mathbf{x}}}{e^{\boldsymbol{\theta}_0^T \mathbf{x}} + e^{\boldsymbol{\theta}_1^T \mathbf{x}}}.$$

Xác suất dự đoán lớp 0:

$$P(y = 0|\mathbf{x}, \boldsymbol{\theta_0}) = \frac{e^{\boldsymbol{\theta}_0^T \mathbf{x}}}{e^{\boldsymbol{\theta}_0^T \mathbf{x}} + e^{\boldsymbol{\theta}_1^T \mathbf{x}}}.$$

Vì tổng xác suất đầu ra của hàm Softmax luôn bằng 1, do đó ta chọn, đơn giản hóa và dễ hiểu hơn, chúng ta có thể chọn $\boldsymbol{\theta}_0 = 0$. Điều này có nghĩa là lớp 0 không có bất kỳ trọng số nào, tức là không có véc tơ tham số nào ảnh hưởng đến lớp 0.

$\boldsymbol{\theta}_0 = 0$, công thức hàm Softmax cho lớp 1 có dạng như sau:

$$P(y = 1|\mathbf{x}, \boldsymbol{\theta_1}) = \frac{e^{\boldsymbol{\theta}_1^T \mathbf{x}}}{e^0 + e^{\boldsymbol{\theta}_1^T \mathbf{x}}} = \frac{e^{\boldsymbol{\theta}_1^T \mathbf{x}}}{1 + e^{\boldsymbol{\theta}_1^T \mathbf{x}}}.$$

Đặt $z = \boldsymbol{\theta}_1^T \mathbf{x}$, thì $e^{\boldsymbol{\theta}_1^T \mathbf{x}} = e^z$. Khi đó, công thức xác suất cho lớp 1 có thể được viết lại như sau:

$$P(y = 1|\mathbf{x}, \boldsymbol{\theta_1}) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}.$$

Đây chính là hàm sigmod của mô hình hồi quy logistic cho lớp 1:

$$P(y = 1|\mathbf{x}, \boldsymbol{\theta_1}) = \sigma(z) = \frac{1}{1 + e^{-z}}.$$

Tương tự, vì tổng xác suất đầu ra bằng 1, do đó xác suất dự đoán lớp 0 là:

$$P(y = 0|\mathbf{x}) = 1 - \sigma(z).$$

Khi chúng ta chọn $\boldsymbol{\theta}_0 = 0$, hàm Softmax với $K = 2$ sẽ tương đương với hàm sigmoid của hồi quy Logistic. Điều này chứng minh rằng hồi quy Softmax với hai lớp có thể quy về hồi quy Logistic.

Kết luận

Chương 2 trình bày về hàm Softmax và mô hình hồi quy Softmax, làm cơ sở lý thuyết để đưa vào ứng dụng thực tiễn. Trong môi trường lập trình, việc tràn số có thể xảy ra khi các giá trị đầu vào quá lớn hoặc quá nhỏ dẫn đến sai sót, việc đưa ra hàm Softmax phiên bản ổn định giúp chúng ta khắc phục được điểm này. Và cuối cùng, với số lớp bằng 2 em đã giải thích tại sao mô hình hồi quy Softmax sẽ trở thành mô hình hồi quy Logistic. Và phương pháp véc tơ hóa các giá trị đầu vào, đầu ra trong quá trình lập trình để rút gọn thời gian, công sức.

Chương 3

Hàm Entropy chéo và tối ưu hàm mất mát

Sau khi đã biết được cách mô hình hồi quy Softmax hoạt động ở Chương 2, việc tiếp theo của chúng ta là phải tìm bộ tham số θ phù hợp nhất cho mỗi lớp để kết quả trả về là chính xác nhất có thể. Chính vì vậy ở chương này sẽ tập trung trình bày xây dựng hàm mất mát entropy chéo, đưa ra công thức đạo hàm, tính toán gradient để thực hiện tối ưu mô hình bằng phương pháp hướng giảm gradient. Bên cạnh đó, các đại lượng liên quan như hàm entropy, độ lệch KL cũng được trình bày, cho thấy mối quan hệ giữa hàm entropy chéo, hàm entropy và độ lệch KL. Và cuối cùng là ý nghĩa của mã hóa one-hot.
Nội dung tham khảo chủ yếu trong [1].

3.1 Hàm entropy chéo

3.1.1 Công thức hàm entropy chéo

Khái niệm entropy chéo (cross entropy) bắt nguồn từ lý thuyết thông tin, được Claude Shannon, người sáng lập lý thuyết thông tin, phát triển vào năm 1948. Trong lý thuyết thông tin và học máy, entropy chéo được sử dụng để đo lường sự khác biệt giữa hai phân phối xác suất. Đây là một công cụ mạnh mẽ giúp tối ưu hóa các mô hình học máy, đặc biệt là trong các bài toán phân loại.

Công thức:

Công thức của cross entropy giữa hai phân phối xác suất P và Q được định nghĩa như sau:

$$H(P,Q) = -\sum_{i=1}^{n} p_i \log(q_i). \tag{3.1}$$

Trong đó:

- $P(x_i) = p_i$ là xác suất thực tế cho kết quả x_i,
- $Q(x_i) = q_i$ là xác suất dự đoán cho kết quả x_i.

Tính chất:

1. $H(P,Q) \geq 0$.

Chứng minh 1:

- p_i và q_i đều là giá trị xác suất, do đó:

$$0 \leq p_i \leq 1 \quad \text{và} \quad 0 \leq q_i \leq 1$$

$$\Rightarrow \log(q_i) \leq 0 \quad \text{do} \quad 0 \leq q_i \leq 1.$$

Vì $p_i \geq 0$ và $\log(q_i) \leq 0$, nên tích $p_i \log(q_i) \leq 0$

Do đó:

$$-\sum_{i=1}^{n} p_i \log(q_i) \geq 0 \Rightarrow H(P,Q) \geq 0.$$

3.1.2 Mối liên hệ giữa hàm entropy chéo, hàm entropy và độ lệch KL

Hàm entropy

Entropy lần đầu tiên được đề xuất bởi Claude Shannon vào năm 1948 trong bài báo mang tính nền tảng của ông, "A Mathematical Theory of Communication". Shannon, được biết đến là cha đẻ của lý thuyết thông tin, đã sử dụng entropy để đo lường mức độ không chắc chắn trong một nguồn thông tin.

Công thức hàm entropy

Entropy của một phân phối xác suất P trên các biến ngẫu nhiên $\{x_1, x_2, \ldots, x_n\}$ với các xác suất tương ứng $\{p_1, p_2, \ldots, p_n\}$ là:

$$H(P) = -\sum_{i=1}^{n} p_i \log(p_i).$$

Nhận xét 3.1 Giá trị entropy chéo giữa hai phân phối P và Q luôn lớn hơn hoặc bằng giá trị entropy của P, tức là:

$$H(P, Q) \geq H(P).$$

Chứng minh.

$$H(p, q) \geq H(p) \Rightarrow \sum_{i=1}^{n} -p_i \log(q_i) \geq \sum_{i=1}^{n} -p_i \log(p_i)$$

$$\Leftrightarrow \sum_{i=1}^{n} p_i \log\left(\frac{1}{q_i}\right) \geq \sum_{i=1}^{n} p_i \log\left(\frac{1}{p_i}\right)$$

$$\Leftrightarrow \sum_{i=1}^{n} p_i \log\left(\frac{1}{p_i}\right) - \sum_{i=1}^{n} p_i \log\left(\frac{1}{q_i}\right) \leq 0$$

$$\Leftrightarrow \sum_{i=1}^{n} p_i \left[\log\left(\frac{1}{p_i}\right) - \log\left(\frac{1}{q_i}\right)\right] \leq 0$$

$$\Leftrightarrow \sum_{i=1}^{n} p_i \log\left(\frac{q_i}{p_i}\right) \leq 0. \quad (*)$$

Ta sẽ chứng minh $(*)$ đúng. Thật vậy ta có $log(x)$ là hàm lõm trên $(0, +\infty)$, áp dụng bất đẳng thức Jensen cho hàm lõm với

$$\sum_{i=1}^{n} p_i = 1.$$

$$\Rightarrow \sum_{i=1}^{n} p_i \log\left(\frac{q_i}{p_i}\right) \leq \log\left(\sum_{i=1}^{n} p_i \left(\frac{q_i}{p_i}\right)\right)$$

$$\Leftrightarrow \log\left[\sum_{i=1}^{n} p_i \frac{q_i}{p_i}\right] = \log\left(\sum_{i=1}^{n} q_i\right) \leq \log\left(\sum_{i=1}^{n} q_i\right) = \log(1) = 0$$

$$\Leftrightarrow \sum_{i=1}^{n} p_i \log\left(\frac{q_i}{p_i}\right) \leq \log\left[\sum_{i=1}^{n} p_i \left(\frac{q_i}{p_i}\right)\right] \leq 0$$

$$\Rightarrow (*) \text{ đúng} \Rightarrow H(P,Q) \geq H(P).$$

□

Độ lệch KL

Độ lệch KL (KL phân kỳ), hay còn gọi là Kullback-Leibler divergence, là một thước đo bất đối xứng giữa hai phân phối xác suất. Nó được đề xuất bởi Solomon Kullback và Richard Leibler trong một bài báo năm 1951.
Độ lệch KL thường được sử dụng trong nhiều lĩnh vực như lý thuyết thông tin, học máy, và xử lý ngôn ngữ tự nhiên để đánh giá mức độ tương đồng giữa phân phối mô hình và phân phối thực tế, với mục tiêu làm cho phân phối dự đoán Q càng gần với phân phối thực tế P càng tốt.

Công thức độ lệch KL

Độ lệch KL giữa hai phân phối xác suất P và Q được định nghĩa như sau:

$$D_{KL}(P\|Q) = \sum_{i=1}^{n} p_i \log\left(\frac{p_i}{q_i}\right).$$

Trong đó

- $P = \{p_1, p_2, \ldots, p_n\}$ là phân phối xác suất thực tế của tập dữ liệu,
- $Q = \{q_1, q_2, \ldots, q_n\}$ là phân phối xác suất dự đoán của mô hình.

Mối liên hệ giữa độ lệch KL, entropy và entropy chéo

Từ công thức Kl-Divergence:

$$D_{KL}(P\|Q) = \sum_{i=1}^{n} p_i \log\left(\frac{p_i}{q_i}\right).$$

Chúng ta có thể tách biểu thức Logarit:

$$D_{KL}(P\|Q) = \sum_{i=1}^{n} p_i \log(p_i) - \sum_{i=1}^{n} p_i \log(q_i).$$

Với

$$H(P) = -\sum_{i=1}^{n} p_i \log(p_i),$$

$$H(P,Q) = -\sum_{i=1}^{n} p_i \log(q_i).$$

Vì vậy, ta sẽ có công thức thể hiện mối liên hệ giữa độ lệch KL, hàm entropy và hàm entropy chéo như sau:

$$D_{KL}(P\|Q) = H(P,Q) - H(P). \tag{3.2}$$

3.2 Xây dựng hàm mất mất và tối ưu hàm mất mát cho mô hình hồi quy Softmax

3.2.1 Mã hóa one-hot

Trong hồi quy Softmax , mỗi điểm dữ liệu được biểu diễn bởi một véc tơ đặc trưng $\mathbf{x}$, và giá trị của nhãn $y = j$ là lớp mà dữ liệu ấy thuộc vào. Kỹ thuật mã hóa one-hot được sử dụng để biểu diễn véc tơ nhãn y dưới dạng véc tơ nhị phân, trong đó mỗi phần tử của véc tơ chỉ có thể nhận một trong số hai giá trị: 0 hoặc 1, trong đó $y_j = 1$, còn lại sẽ bằng 0. Ngoài ra, đưa về dạng mã hóa one hot sẽ giúp dễ dàng tính hàm entropy chéo. Bởi vì chỉ có thành phần bằng 1 sẽ được tính trong công thức, còn lại bằng 0 thì sẽ bỏ qua.

Giả sử chúng ta có K lớp khác nhau trong bài toán phân loại. Một điểm dữ liệu $\mathbf{x}_i$ thuộc vào lớp j. véc tơ $\mathbf{y}_i$ được biểu diễn dưới dạng mã hóa one-hot dưới dạng một véc tơ cột có độ dài K, trong đó chỉ có một phần tử ở vị trí thứ j là 1, và các phần tử còn lại là 0.

$$\mathbf{y}_i = (0, 0, \ldots, 1, \ldots, 0)^T.$$

Trong đó, chỉ có phần tử thứ j là 1, tức là $y_{ij} = 1$, với $1 \leq j \leq K$, và các phần tử còn lại $y_{ik} = 0$, với $k \neq j$.

Ví dụ: có $K = 3$ lớp cần phân loại, giả sử điểm dữ liệu thứ nhất $\mathbf{x}_1$ thuộc vào lớp 2 tức là nhãn của nó $y_1 = 2$ thì véc tơ one hot là: $\mathbf{y}_1 = [0, 1, 0]$

Tổng quát: Với bộ dữ liệu có N điểm, véc tơ xác suất thực tế của N điểm là $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$ và có K lớp thì dạng mã hóa one hot của véc tơ $\mathbf{y}$ là ma trận $\mathbf{Y} \in \mathbb{R}^{K \times N}$:

$$\mathbf{Y} = \begin{pmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \ldots & \mathbf{y}_N \end{pmatrix}.$$

Trong đó, mỗi cột $\mathbf{y}_i$ là dạng mã hóa one hot xác suất thực tế của điểm dữ liệu thứ i.

Ví dụ: Có $K = 3$ lớp cần phân loại, 4 điểm dữ liệu và véc tơ xác suất thực tế $\mathbf{y} = (1, 2, 1, 3)^T$.

Ma trận one hot Y sẽ có dạng:

$$\mathbf{Y} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

3.2.2 Xây dựng hàm mất mát

Trong mô hình hồi quy Softmax, mục tiêu của chúng ta là làm cho xác suất dự đoán đầu ra Q gần với xác suất thực tế P nhất có thể. Điều này đồng nghĩa với việc làm giá trị của độ lệch KL $D_{KL}(P\|Q)$ nhỏ nhất.

Từ công thức thể hiện mối liên hệ giữa độ lệch KL, hàm entropy và hàm entropy chéo, ta nhận thấy $H(P)$ là entropy của phân phối xác suất thực tế P, chỉ phụ thuộc vào xác suất thực tế (luôn cho trước). Do đó, $H(P)$ luôn là một hằng số. Vì vậy, trong quá trình huấn luyện mô hình hồi quy Softmax, thay vì trực tiếp tối ưu hàm độ lệch KL, chúng ta sử dụng hàm mất mát theo hàm entropy chéo $H(P, Q)$ để tìm giá trị tối ưu của bộ tham số.

Nhận xét: Hàm entropy chéo đạt giá trị nhỏ nhất khi và chỉ khi hai phân phối $P = Q$ hay $p_i = q_i$ với mọi i.

Chứng minh: Từ tính chất $H(P, Q) \geq H(P)$ và kết hợp với điều kiện $P = Q$, ta có:

$$H(P, Q) = H(P, P) = -\sum_{i=1}^{n} p_i \log(p_i) = H(P).$$

Hàm mất mát entropy chéo

Cho một điểm dữ liệu $(\mathbf{x}_i, \mathbf{y}_i)$ trong đó $\mathbf{x}_i$ là véc tơ đặc trưng, $\mathbf{y}_i$ là véc tơ xác suất thực tế ở dạng mã hóa one-hot $\mathbf{x}_i$ và véc tơ xác suất dự đoán $\hat{\mathbf{y}} = (\hat{y}_{i1}, \hat{y}_{i2}, \ldots, \hat{y}_{iK})^T$ ta sẽ có hàm mất mát cho điểm dữ liệu $(\mathbf{x}_i, \mathbf{y}_i)$ như sau.

$$J(\boldsymbol{\Theta}; x_i, y_i) = -\sum_{j=1}^{K} y_{ij} \log(\hat{y}_{ij}). \tag{3.3}$$

Tổng quát: Hàm mất mát tổng quát cho toàn bộ tập dữ liệu (với N mẫu), ta sẽ lấy giá trị trung bình của giá trị entropy chéo:

$$J(\boldsymbol{\Theta}; \mathbf{X}, \mathbf{Y}) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{K} y_{ij} \log(\hat{y}_{ij}). \tag{3.4}$$

Ví dụ: Với $N = 4$ điểm dữ liệu, $K = 3$ lớp, giả sử ta có ma trận xác suất thực tế dưới dạng mã hóa one-hot và ma trận xác suất dự đoán như sau:

Ma trận xác suất thực tế (one-hot encoding):

$$Y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Ma trận dự đoán $\hat{Y}$:

$$\hat{Y} = \begin{bmatrix} 0.3792 & 0.3072 & 0.4263 & 0.2668 \\ 0.3104 & 0.4147 & 0.2248 & 0.2978 \\ 0.3104 & 0.2780 & 0.3490 & 0.4354 \end{bmatrix}.$$

1. Đối với điểm dữ liệu thứ nhất.

- Xác suất thực tế $\mathbf{y}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ - Xác suất dự đoán $\hat{\mathbf{y}}_1 = \begin{bmatrix} 0.3792 \\ 0.3104 \\ 0.3104 \end{bmatrix}$.

$$\Rightarrow J_1 = -(1 \cdot \log(0.3792) + 0 \cdot \log(0.3104) + 0 \cdot \log(0.3104)) = -\log(0.3792) \approx 0.9696.$$

2. Đối với điểm dữ liệu thứ hai.

- Xác suất thực tế $\mathbf{y}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ - Xác suất dự đoán $\hat{\mathbf{y}}_2 = \begin{bmatrix} 0.3072 \\ 0.4147 \\ 0.2780 \end{bmatrix}$.

$$\Rightarrow J_2 = -(0 \cdot \log(0.3072) + 1 \cdot \log(0.4147) + 0 \cdot \log(0.2780)) = -\log(0.4147) \approx 0.8802.$$

3. Đối với điểm dữ liệu thứ ba.

- Xác suất thực tế $\mathbf{y}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ - Xác suất dự đoán $\hat{\mathbf{y}}_3 = \begin{bmatrix} 0.4263 \\ 0.2248 \\ 0.3490 \end{bmatrix}$.

$$\Rightarrow J_3 = -(0 \cdot \log(0.4263) + 0 \cdot \log(0.2248) + 1 \cdot \log(0.3490)) = -\log(0.3490) \approx 1.0527.$$

4. Đối với điểm dữ liệu thứ tư.

- Xác suất thực tế $\mathbf{y}_4 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ - Xác suất dự đoán $\hat{\mathbf{y}}_4 = \begin{bmatrix} 0.2668 \\ 0.2978 \\ 0.4354 \end{bmatrix}$.

$\Rightarrow J_4 = -(0 \cdot \log(0.2668) + 0 \cdot \log(0.2978) + 1 \cdot \log(0.4354)) = -\log(0.4354) \approx 0.8315.$

$$\Rightarrow J = \frac{1}{4}(J_1 + J_2 + J_3 + J_4) = \frac{1}{4}(0.9696 + 0.8802 + 1.0527 + 0.8315) = 0.9335.$$

.

3.2.3 Tối ưu hàm mất mát

[1] Với một điểm dữ liệu $(\mathbf{x}_i, \mathbf{y}_i)$, ta đặt

$$J_i(\theta) = -\sum_{j=1}^{K} y_{ij} \log\left(\frac{e^{\theta_j^T \mathbf{x}_i}}{\sum_{k=1}^{K} e^{\theta_k^T \mathbf{x}_i}}\right)$$

$$= -\sum_{j=1}^{K}\left(y_{ij}\theta_j^T \mathbf{x}_i - y_{ij} \log\left(\sum_{k=1}^{K} e^{\theta_k^T \mathbf{x}_i}\right)\right) \quad \textbf{Vì:} \ \left(\sum_{j=1}^{K} y_{ij} = 1\right)$$

$$= -\sum_{j=1}^{K} y_{ij}\theta_j^T \mathbf{x}_i + \log\left(\sum_{k=1}^{K} e^{\theta_k^T \mathbf{x}_i}\right). \quad (**)$$

Với véc tơ Gradient của véc tơ trọng số θ^j của lớp thứ j với điểm dữ liệu thứ i được định nghĩa như sau

$$\frac{\partial J_i(\theta)}{\partial \theta_j} = \left[\frac{\partial J_i(\theta)}{\partial \theta_{j1}}, \frac{\partial J_i(\theta)}{\partial \theta_{j2}}, \ldots, \frac{\partial J_i(\theta)}{\partial \theta_{jn}}\right]. \tag{3.5}$$

Trong đó $\frac{\partial J_i(\theta)}{\partial \theta_{jk}}$ là đạo hàm riêng của thành phần thứ k trong của véc tơ trọng số θ_j của lớp j. Với $0 \leq k \leq n$

Và từ $(**)$

$$\Rightarrow \frac{\partial J(\theta)_i}{\partial \theta_{jk}} = -y_{ij}x_k + \frac{e^{\theta_j^T \mathbf{x}_i}}{\sum_{c=1}^{K} e^{(\theta_c)^T \mathbf{x}^i}} x_k$$

$$= -y_{ij}x_k + \hat{y}_{ij}x_k$$

$$= (\hat{y}_{ij} - y_{ij})x_k.$$

Kết hợp với (2.5)

$$\Rightarrow \frac{\partial J_i(\theta)}{\partial \theta_j} = [(\hat{y}_{ij} - y_{ij})x_1, (\hat{y}_{ij} - y_{ij})x_2, \ldots, (\hat{y}_{ij} - y_{ij})x_m]$$

$$= (\hat{y}_{ij} - y_{ij})\,[x_1, x_2, \ldots, x_m] = (\hat{y}_{ij} - y_{ij})\mathbf{x}_i.$$

Đặt $(\hat{y}_{ij} - y_{ij}) = e_{ij}$, ta thu được véc tơ gradient $\frac{\partial J_i(\theta)}{\partial \theta_j} \in \mathbb{R}^n$ của bộ tham số θ_j của lớp thứ j ứng với điểm dữ liệu $\mathbf{x}_i$:

$$\frac{\partial J_i(\theta)}{\partial \theta_j} = e_{ij}\mathbf{x}_i. \tag{3.6}$$

Kết hợp (2.6) và với K lớp, ta sẽ thu được một ma trận $\frac{\partial J(\theta)_i}{\partial \theta} \in \mathbb{R}^{n\times K}$ sẽ là một ma trận mà mỗi cột chính là một véc tơ gradient

$$\frac{\partial J_i(\theta)}{\partial \theta} = \mathbf{x}_i[e_{i1}, e_{i2}, \ldots, e_{iK}] = \mathbf{x}_i\mathbf{e}_i^T. \tag{3.7}$$

Trong đó $\mathbf{e}_i$ được tính như sau:

$$\mathbf{e}_i = \hat{\mathbf{y}}_i - \mathbf{y}_i = (\hat{y}_{i1}, \hat{y}_{i2}, \ldots, \hat{y}_{iK})^T - (y_{i1}, y_{i2}, \ldots, y_{iK})^T = (e_{i1}, e_{i2}, \ldots, e_{iK})^T.$$

.

Nhật xét: Với công thức tính Gradient của θ ở công thức (2.7) ta sẽ dùng phương pháp hướng giảm gradient. Với η là hệ số học, ta có công thức cập nhật Θ như sau:

$$\Theta := \Theta - \eta\mathbf{x}_i\mathbf{e}^T. \tag{3.8}$$

Kết luận

Kết thúc Chương 3, chúng ta đã nắm rõ phương pháp đạo hàm, tính gradient của hàm mất mát entropy chéo để từ đó tối ưu hàm mất mát bằng phương pháp hướng giảm gradient, trả về kết quả nghiệm là một bộ tham số θ phù hợp nhất cho mô hình. Ngoài ra, việc mã hóa one-hot cho chúng ta thấy được lợi ích khi tính toán hàm mất mát trong quá trình lập trình.

Chương 4

Ứng dụng hồi quy Softmax vào bài toán nhận dạng chữ cái viết tay

Tóm tắt

Sau khi đã đi qua hết các Chương 1, 2 và 3, chúng ta nắm được cơ sở lý thuyết, cách mô hình hoạt động, phương pháp tối ưu hàm mất mát, thì ở chương này em sẽ trình bày kết quả lập trình thực nghiệm với bộ dữ liệu thực tế là bộ dữ liệu *letter* để phân loại 26 chữ cái viết tay và đánh giá và so sánh các mô hình thông qua các chỉ số đánh giá đã giới thiệu ở Chương 1.

4.1 Giới thiệu bộ dữ liệu *letter*

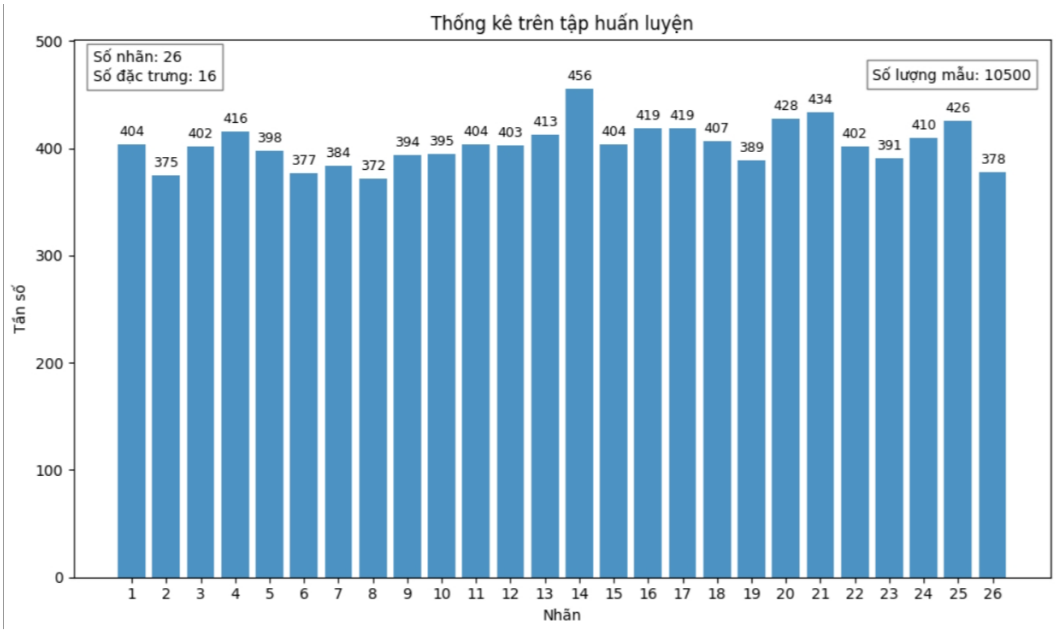
4.1.1 Tổng quan bộ dữ liệu *letter*

Bộ dữ liệu *letter* [4] là bộ dữ liệu của các bản chữ cái viết tay, bao gồm 26 lớp tương ứng với 26 chữ cái, và 16 đặc trưng cho mỗi dữ liệu. Bộ dữ liệu được trích rút từ thư viện *LIBSVM* [4]. Bộ dữ liệu đã được tác giả làm sạch và chuẩn hóa dưới dạng số học để ta có thể dễ dàng sử dụng trong việc huấn luyện và đánh giá mô hình.

Bộ dữ liệu đã được chia thành 3 tập là tập huấn luyện (training set), kiểm thử (testing set) và tập thẩm định (validation set).

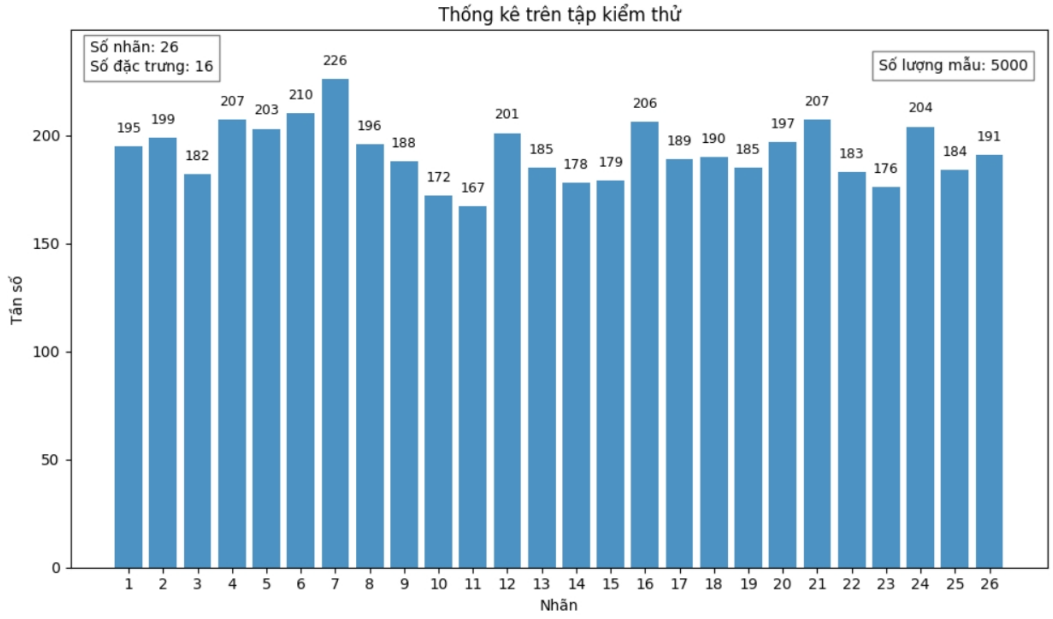
4.1.2 Thống kê bộ dữ liệu

Thống kê trên tập huấn luyện



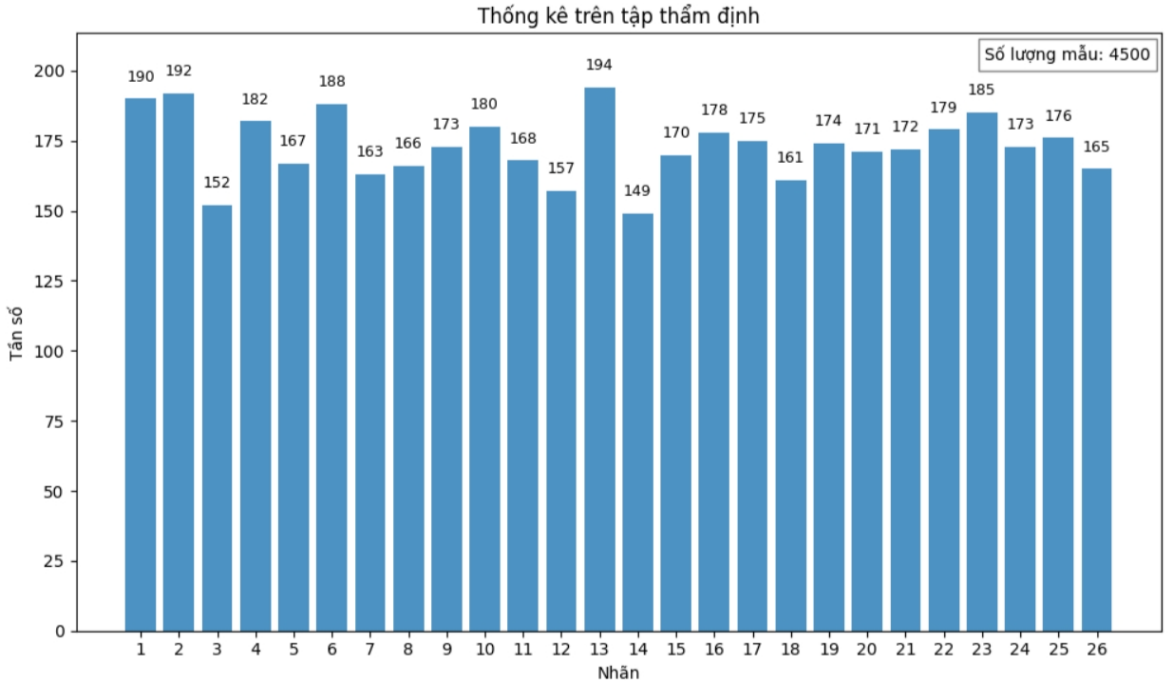
Hình 4.1: Biểu đồ thống kê của tập huấn luyện

Thống kê trên tập kiểm thử



Hình 4.2: Biểu đồ thống kê của tập kiểm thử

Thống kê trên tập thẩm định



Hình 4.3: Biểu đồ thống kê của tập thẩm định

4.2 Huấn luyện mô hình bằng thuật toán hướng giảm gradient

4.2.1 Huấn luyện mô hình với cỡ bước hằng

Giá trị của các siêu tham số trong quá trình huấn luyện

- Cỡ bước: $\eta = 0.03$.
- Sai số cho trước: $\varepsilon = 10^{-8}$.
- Số lần lặp tối đa: 500000.

```python
def softmax_regression(X, y, theta_init, eta, tol=1e-8, max_count=500000):
    theta = theta_init
    C = theta.shape[1]
    Y = convert_labels(y, C)
    N = X.shape[1]
    d = X.shape[0]

    count = 0
    prev_theta = theta

    while count < max_count:
        # Shuffle data
        mix_id = np.random.permutation(N)
        for i in mix_id:
            xi = X[:, i].reshape(d, 1)
            yi = Y[:, i].reshape(C, 1)
            ai = softmax(np.dot(theta.T, xi))
            grad = xi.dot((ai - yi).T)
            theta_new = theta - eta * grad
            count += 1

            # Check for convergence
            if np.linalg.norm(theta_new - prev_theta) < tol:
                return theta_new

            prev_theta = theta
            theta = theta_new

    return theta
```

Hình 4.4: Huấn luyện với cỡ bước hằng

Ma trận θ thu được

```
[40]: 1 print(theta)
```

```
[[ 1.46004762e+00 -4.22658822e+00  1.85685835e+00 -6.89425891e+00
  -3.67404860e+00 -2.42826724e+00  3.26950605e-01 -4.06521021e+00
   3.86461329e+00  3.54895673e-01  2.98146079e-02  1.06697836e-01
   2.51980962e+00 -9.69485758e-01 -2.02217490e+00  1.60300755e+00
   3.11234792e+00 -1.79070702e+00 -1.18838130e-01  2.79926040e+00
  -8.57075998e-02  1.96336099e+00  4.35365295e+00 -7.06084553e-01
   3.91883488e+00 -1.24898939e+00]
 [-2.44210354e+00  2.86911018e+00  1.79872012e+00  5.38492581e+00
   1.57980840e+00 -4.20705394e-01  1.10525379e+00  1.35431554e+00
  -3.20864355e-01 -3.06731242e+00 -8.86045878e-01  1.13772535e+00
  -2.46270866e-01  3.27624331e-01  2.29336936e+00 -2.81361085e+00
  -2.14699187e+00  1.27760521e+00 -2.60117842e-01  1.67873011e+00
   2.78275304e+00  1.08980655e+00 -1.73763698e+00 -8.05192968e-01
  -2.08208503e+00 -3.64427189e+00]
 [ 2.59849633e+00 -1.93248173e+00 -2.69773633e+00 -2.18033014e+00
  -3.47638434e+00 -4.55686965e-02  1.98659608e+00  9.39232500e+00
  -1.14953763e+01 -3.95068558e+00  4.93544628e+00 -3.32175790e+00
```

Hình 4.5: Ma trận θ thu được.

Kết quả trên tập huấn luyện với cỡ bước hằng

Kết quả của độ chính xác (accuracy) trên tập huấn luyện là 75.73%

.

```python
# Dự đoán trên tập huấn luyện
Y_train_pred = predict(theta, X_train)
print("Các lớp được dự đoán cho tập huấn luyện với 20 dữ liệu đầu:\n", Y_train_pred[:20])
print("Các lớp thực sự cho tập huấn luyện với 20 dữ liệu đầu:\n", y_train[:20])
acc_train = accuracy_score(y_train.T, Y_train_pred)
print(f"Độ chính xác cho tập huấn luyện: {acc_train:.2f}%")
```

```
Các lớp được dự đoán cho tập huấn luyện với 20 dữ liệu đầu:
 [21 11  8  2 21 10  9  3  7  6  9  4  3  2 21 18 24 11 23 19]
Các lớp thực sự cho tập huấn luyện với 20 dữ liệu đầu:
 [21 11  8  2 21 10  9  3  7  6  9  2  3  6 21  0 24 11 23 19]
Độ chính xác cho tập huấn luyện: 75.73%
```

Hình 4.6: Độ chính xác trên tập huấn luyện.

Kết quả trên tập kiểm thử

Kết quả độ chính xác trên tập kiểm thử với cỡ bước hằng là 75.02%.

```python
# Dự đoán trên tập kiểm thử
Y_test_pred = predict(theta, X_test)
print("Các lớp được dự đoán cho tập kiểm thử cho 20 dữ liệu đầu:\n", Y_test_pred[:20])
print("Các lớp thực sự cho tập kiểm thử cho 20 dữ liệu đầu:\n", y_test[:20])

acc_test = accuracy_score(y_test.T, Y_test_pred)
print(f"Độ chính xác cho tập kiểm thử: {acc_test:.2f}%")

```

```
Các lớp được dự đoán cho tập kiểm thử cho 20 dữ liệu đầu:
 [16 17  5  5 19 19 16  9  2 14 21  8  7 17  4 16  5 23  5  4]
Các lớp thực sự cho tập kiểm thử cho 20 dữ liệu đầu:
 [16 17  5  5 19 19  6 25  2 14 21  8  7  3  4 16 15 23  5  4]
Độ chính xác cho tập kiểm thử: 75.02%
```

Hình 4.7: Độ chính xác trên tập kiểm thử.

Bảng 4.1: Các chỉ số đánh giá cho từng lớp với cỡ bước hằng trên tập kiểm thử

Label	TP	FP	FN	Precision	Recall	F1-score
1	176	48	19	0.79	0.90	0.84
2	136	65	63	0.68	0.68	0.68
3	136	51	46	0.73	0.75	0.74
4	157	54	50	0.74	0.76	0.75
5	163	65	40	0.71	0.80	0.76
6	155	38	55	0.80	0.74	0.77

Bảng 4.1: (tiếp theo)

Label	TP	FP	FN	Precision	Recall	F1-score
7	109	56	117	0.66	0.48	0.56
8	100	63	96	0.61	0.51	0.56
9	146	29	42	0.83	0.78	0.80
10	137	33	35	0.81	0.80	0.80
11	121	88	46	0.58	0.72	0.64
12	158	29	43	0.84	0.79	0.81
13	169	19	16	0.90	0.91	0.91
14	148	30	30	0.83	0.83	0.83
15	100	46	79	0.68	0.56	0.62
16	169	32	37	0.84	0.82	0.83
17	141	57	48	0.71	0.75	0.73
18	134	62	56	0.68	0.71	0.69
19	109	91	76	0.55	0.59	0.57
20	161	47	36	0.77	0.82	0.80
21	180	35	27	0.84	0.87	0.85
22	164	37	19	0.82	0.90	0.85
23	158	28	18	0.85	0.90	0.87
24	151	47	53	0.76	0.74	0.75
25	132	43	52	0.75	0.72	0.74
26	141	56	50	0.72	0.74	0.73

Kết quả trên tập thẩm định

Kết quả độ chính xác đánh giá trên tập thẩm định là 75,47%.

Metric	Value
Precision trung bình có trọng số	0.75
Recall trung bình có trọng số	0.76
F1-score trung bình có trọng số	0.75
Accuracy	75.06%

Bảng 4.2: Các chỉ số đánh giá mô hình trên tập dữ liệu kiểm thử

```python
# Dự đoán trên tập thẩm định
Y_val_pred = predict(theta, X_val)
print("Các lớp được dự đoán cho tập thẩm định cho 20 dữ liệu đầu:\n", Y_val_pred[:20])
print("Các lớp thực sự cho tập thẩm định cho 20 dữ liệu đầu:\n", y_val[:20])

acc_test = accuracy_score(y_val, Y_val_pred.T)
print(f"Độ chính xác cho tập thẩm định: {acc_test:.2f}%")

```

```
Các lớp được dự đoán cho tập thẩm định cho 20 dữ liệu đầu:
 [ 7  5  2 10  9 11  0  1 17 25 25  1 15 11 13 17  5 12  4 19]
Các lớp thực sự cho tập thẩm định cho 20 dữ liệu đầu:
 [13  5  6 17  9 11  0  1 18 25 18  1 15 11 13 17  5 12  4 19]
Độ chính xác cho tập thẩm định: 75.47%
```

Hình 4.8: Độ chính xác trên tập kiểm thử.

Bảng 4.3: Các chỉ số đánh giá cho từng lớp với cỡ bước hằng trên tập thẩm định

Label	TP	FP	FN	Precision	Recall	F1-score
1	163	42	27	0.80	0.86	0.83
2	146	74	46	0.66	0.76	0.71
3	112	35	40	0.76	0.74	0.75
4	146	46	36	0.76	0.80	0.78
5	132	53	35	0.71	0.79	0.75
6	129	31	59	0.81	0.69	0.74
7	81	61	82	0.57	0.50	0.53
8	72	59	94	0.55	0.43	0.48
9	136	21	37	0.87	0.79	0.82
10	150	20	30	0.88	0.83	0.86

Bảng 4.3: (tiếp theo)

Nhãn	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F1-score**
11	116	77	52	0.60	0.69	0.64
12	117	25	40	0.82	0.75	0.78
13	172	12	22	0.93	0.89	0.91
14	131	20	18	0.87	0.88	0.87
15	109	45	61	0.71	0.64	0.67
16	146	38	32	0.79	0.82	0.81
17	118	54	57	0.69	0.67	0.68
18	115	58	46	0.66	0.71	0.69
19	98	88	76	0.53	0.56	0.54
20	135	31	36	0.81	0.79	0.80
21	150	25	22	0.86	0.87	0.86
22	163	38	16	0.81	0.91	0.86
23	168	28	17	0.86	0.91	0.88
24	137	55	36	0.71	0.79	0.75
25	131	33	45	0.80	0.74	0.77
26	123	35	42	0.78	0.75	0.76

Metric	**Value**
Precision trung bình có trọng số	0.76
Recall trung bình có trọng số	0.75
F1-score trung bình có trọng số	0.75
Accuracy	75.47%

Bảng 4.4: Các chỉ số đánh giá mô hình trên tập thẩm định.

4.2.2 Huấn luyện với hướng giảm gradient và thuật toán quay lui

Các giá trị của siêu tham số.

- Cỡ bước hay tốc độc học: Khởi đầu $t_k = 1$.
- $m = 0.01$.
- $\alpha = 0.5$.
- Số epochs = 256.
- Kích thước Batch trong mỗi Epoch : 1024.
- Sai số: $\varepsilon = 10^{-8}$

```python
def backtracking_softmax(X_train, y_train1hot, eta=1, m=0.5, alpha=0.5, batch_size=1024, epochs=256):
    d, C = X_train.shape[0], y_train1hot.shape[0]
    theta_init = np.random.randn(d, C)
    theta = [theta_init]

    for epoch in range(epochs):
        for i in range(0, X_train.shape[1], batch_size):
            X_batch = X_train[:, i:i+batch_size]
            y_batch = y_train1hot[:, i:i+batch_size]

            ai = softmax_stable(np.dot(theta[-1].T, X_batch))
            grad = np.dot(X_batch, (ai - y_batch).T) / batch_size

            theta_new = theta[-1] - eta * grad
            t_k = eta
            j = 0
            while (loss(X_train, y_train1hot, theta_new) > \
                  loss(X_train, y_train1hot, theta[-1]) - m * t_k * np.linalg.norm(grad) ** 2) and (j < 5):
                t_k *= alpha
                theta_new = theta[-1] - t_k * grad
                j += 1

            theta.append(theta_new)
            if np.linalg.norm(theta[-1] - theta[-2]) < 1e-8:
                break

        print(f"Epoch {epoch + 1} complete")
        print(f"Loss: {loss(X_train, y_train1hot, theta[-1])}")

    theta_final = theta[-1]
    return theta_final
```

Hình 4.9: Thuật toán quay lui .

Kết quả trên tập huấn luyện

Độ chính xác trên tập huấn luyện với thuật toán quay lui là 72.65%.

```
# Dự đoán trên tập huấn luyện
Y_train_pred = predict(theta_final, X_train)
print("Các lớp được dự đoán cho tập huấn luyện với 20 dữ liệu đầu:\n", Y_train_pred[:30])
print("Các lớp thực sự cho tập huấn luyện với 20 dữ liệu đầu:\n", y_train[:30])

acc_train = accuracy_score(y_train.T, Y_train_pred)
print(f"Độ chính xác cho tập huấn luyện: {acc_train:.2f}%")

```

```
Các lớp được dự đoán cho tập huấn luyện với 20 dữ liệu đầu:
 [25  4 17  7 23  5 13 17 12  3 21  9  1  3 17  7  4 25  7 16 14 11 21 23
  7 12 24  3  8 13]
Các lớp thực sự cho tập huấn luyện với 20 dữ liệu đầu:
 [25 15 18  7  7  5 13 17 12  3 21  0 10  3 18  7  4 25 14 16 16 11 21 23
 21 12 24  3  4 13]
Độ chính xác cho tập huấn luyện: 72.65%
```

Hình 4.10: Kết quả độ chính xác trên tập huấn luyện với thuật toán quay lui.

Kết quả trên tập kiểm thử

Độ chính xác của thuật toán quay lui trên tập kiểm thử là 72.92%.

```
# Dự đoán trên tập kiểm thử
Y_test_pred = predict(theta_final, X_test)
print("Các lớp được dự đoán cho tập kiểm thử với 20 dữ liệu đầu:\n", Y_test_pred[:20])
print("Các lớp thực sự cho tập kiểm thử với 20 dữ liệu đầu:\n", y_train[:20])
acc_test = accuracy_score(y_test.T, Y_test_pred)
print(f"Độ chính xác cho tập kiểm thử: {acc_test:.2f}%")

```

```
Các lớp được dự đoán cho tập kiểm thử với 20 dữ liệu đầu:
 [14 17  5  5 19 19  1  5  2 14 21  8  7 17  4 18  5 23  5  4]
Các lớp thực sự cho tập kiểm thử với 20 dữ liệu đầu:
 [25 15 18  7  7  5 13 17 12  3 21  0 10  3 18  7  4 25 14 16]
Độ chính xác cho tập kiểm thử: 72.92%
```

Hình 4.11: Kết quả độ chính xác trên tập kiểm thử với thuật toán quay lui.

Bảng 4.5: Các chỉ số đánh giá cho từng lớp với thuật toán quay lui trên tập kiểm thử

Label	TP	FP	FN	Precision	Recall	F1-score
1	176	62	19	0.74	0.90	0.81
2	139	65	60	0.68	0.70	0.69
3	140	54	42	0.72	0.77	0.74
4	154	60	53	0.72	0.74	0.73
5	158	55	45	0.74	0.78	0.76
6	150	46	60	0.77	0.71	0.74
7	99	59	127	0.63	0.44	0.52

Bảng 4.5: (tiếp theo)

Label	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F1-score**
8	72	41	124	0.64	0.37	0.47
9	140	34	48	0.80	0.74	0.77
10	136	33	36	0.80	0.79	0.80
11	116	96	51	0.55	0.69	0.61
12	149	25	52	0.86	0.74	0.79
13	168	27	17	0.86	0.91	0.88
14	142	36	36	0.80	0.80	0.80
15	100	50	79	0.67	0.56	0.61
16	168	43	38	0.80	0.82	0.81
17	137	73	52	0.65	0.72	0.69
18	144	74	46	0.66	0.76	0.71
19	102	115	83	0.47	0.55	0.51
20	152	39	45	0.80	0.77	0.78
21	177	19	30	0.90	0.86	0.88
22	167	39	16	0.81	0.91	0.86
23	159	45	17	0.78	0.90	0.84
24	143	65	61	0.69	0.70	0.69
25	117	45	67	0.72	0.64	0.68
26	141	54	50	0.72	0.74	0.73

Chỉ số đánh giá mô hình	**Giá trị**
Accuracy	72.92%
Precision trung bình có trọng số	0.73
Recall trung bình có trọng số	0.73
F1-score trung bình có trọng số	0.73

Bảng 4.6: Chỉ số đánh giá mô hình với thuật toán quay lui trên tập kiểm thử

Kết quả trên tập thẩm định với thuật toán quay lui

Độ chính xác trên tập thẩm định với thuật toán quay lui cho độ chính xác 72.98%.

```
# Dự đoán trên tập thẩm định
Y_val_pred = predict(theta_final, X_val)
print("Các lớp được dự đoán cho tập thẩm định với 20 dữ liệu đầu:\n", Y_val_pred[:20])
print("Các lớp thực sự cho tập thẩm định với 20 dữ liệu đầu:\n", y_val[:20])

acc_val = accuracy_score(y_val.T, Y_val_pred)
print(f"Độ chính xác cho tập thẩm định: {acc_val:.2f}%")

```

```
Các lớp được dự đoán cho tập thẩm định với 20 dữ liệu đầu:
 [13  5  2 17  9 11  0  1 17 25 25  1 15 11 12 17  5 12  4 19]
Các lớp thực sự cho tập thẩm định với 20 dữ liệu đầu:
 [13  5  6 17  9 11  0  1 18 25 18  1 15 11 13 17  5 12  4 19]
Độ chính xác cho tập thẩm định: 72.98%
```

Hình 4.12: Độ chính xác trên tập thẩm định với thuật toán quay lui .

Chỉ số đánh giá mô hình	**Giá trị**
Accuracy	72.98%
Precision trung bình có trọng số	0.73
Recall trung bình có trọng số	0.73
F1-score trung bình có trọng số	0.73

Bảng 4.7: Chỉ số đánh giá mô hình với thuật toán quay lui trên tập thẩm định

Bảng 4.8: Kết quả độ chính xác của thuật toán quay lui trên tập thẩm định

Label	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F1-score**
1	160	50	30	0.76	0.84	0.80
2	146	80	46	0.65	0.76	0.70
3	115	41	37	0.74	0.76	0.75
4	145	53	37	0.73	0.80	0.76
5	119	41	48	0.74	0.71	0.73
6	119	39	69	0.75	0.63	0.69
7	69	60	94	0.53	0.42	0.47
8	55	35	111	0.61	0.33	0.43

Bảng 4.8: (tiếp theo)

Label	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F1-score**
9	135	38	38	0.78	0.78	0.78
10	149	18	31	0.89	0.83	0.86
11	108	68	60	0.61	0.64	0.63
12	112	22	45	0.84	0.71	0.77
13	170	30	24	0.85	0.88	0.86
14	123	31	26	0.80	0.83	0.81
15	107	39	63	0.73	0.63	0.68
16	148	44	30	0.77	0.83	0.80
17	122	58	53	0.68	0.70	0.69
18	120	78	41	0.61	0.75	0.67
19	85	111	89	0.43	0.49	0.46
20	128	23	43	0.85	0.75	0.80
21	148	17	24	0.90	0.86	0.88
22	157	37	22	0.81	0.88	0.84
23	166	42	19	0.80	0.90	0.84
24	132	74	41	0.64	0.76	0.70
25	117	40	59	0.75	0.66	0.70
26	129	47	36	0.73	0.78	0.76

4.2.3 So sánh và đánh giá

Bảng 4.9: Bảng so sánh các chỉ số đánh giá mô hình trên tập kiểm thử

Mô hình	**Cỡ bước hằng**	**Thuật toán quay lui**
Accuracy	75.06%	72.92%
Precision trung bình có trọng số	0.75	0.73
Recall trung bình có trọng số	0.76	0.73
F1-score có trọng số	0.75	0.73

Bảng 4.10: Bảng so sánh các chỉ số đánh giá mô hình trên tập thẩm định

Mô hình	**Cỡ bước hằng**	**Thuật toán quay lui**
Accuracy	75.47%	72.98%
Precision trung bình có trọng số	0.76	0.73
Recall trung bình có trọng số	0.76	0.73
F1-score có trọng số	0.75	0.73

Tổng quan về hai bảng so sánh chỉ số đánh giá mô hình trên tập kiểm thử và tập thẩm định cho thấy mô hình đang hoạt động ổn định và có khả năng tổng quát hóa tốt, với cỡ bước hằng kết quả cho tốt hơn khoảng 2% - 3% so với thuật toán quay lui. Đầu tiên, về độ chính xác (Accuracy), mô hình với cỡ bước hằng đạt khoảng 75.06% trên tập kiểm thử và 75.47% trên tập thẩm định và đều cao hơn so với thuật toán quay lui, nhưng không đáng kể. Đây là mức độ chính xác khá cao, cho thấy khả năng phân loại ổn định và chính xác của mô hình trên các điểm dữ liệu.

Tiếp theo, các chỉ số Precision, Recall, và F1-score cũng cho thấy mô hình có hiệu suất tương đối tốt. Với cỡ bước hằng, Precision trung bình có trọng số ở mức 0.75 trên tập kiểm thử và 0.76 trên tập thẩm định, chỉ ra tỉ lệ dự đoán chính xác của mô hình. Recall trung bình có trọng số đạt mức 0.76 trên cả hai tập dữ liệu, cho thấy khả năng phát hiện đúng các trường hợp dương tính. F1 Score có trọng số, một chỉ số kết hợp giữa Precision và Recall, cũng ổn định ở

mức khoảng 0.75 cho cả hai tập dữ liệu. Tất cả các chỉ số này với thuật toán quay lui đều nhỏ hơn so với cỡ bước hằng, tuy nhiên là không đáng kể

So sánh giữa các chỉ số trên hai tập dữ liệu và 2 mô hình cỡ bước hằng và thuật toán quay lui không cho thấy sự khác biệt quá lớn, cho thấy mô hình tổng quát hóa tốt từ tập huấn luyện sang tập thẩm định. Mặc dù có sự dao động nhỏ về Accuracy và các chỉ số khác, nhưng sự khác biệt này không ảnh hưởng đáng kể đến độ tin cậy của mô hình.

Sự chênh lệch trong hiệu suất của mô hình giữa hai mô hình cỡ bước hằng và thuật toán quay lui (quy tắc Armijo) có thể được hiểu là do kích thước bộ dữ liệu huấn luyện không quá lớn, mô hình có thể không học được đầy đủ các đặc trưng quan trọng, ngoài ra có thể do sự đa dạng của dữ liệu trong tập dữ liệu, tần suất xuất hiện các lớp khác biệt lớn. Sự lựa chọn các siêu tham số như số lần lặp, sai số, epoch, batch,...chưa thực sự tốt, ngoài ra bộ tham số θ khởi tạo ban đầu cũng ảnh hưởng đến kết quả cuối cùng của mô hình.

Để cải thiện mô hình chúng ta có thể kích thước dữ liệu huấn luyện bằng cách thu thập thêm dữ liệu hoặc sinh dữ liệu dựa trên phân phối, mô hình có thể học được nhiều đặc trưng hơn và tổng quát hóa tốt hơn trên các tập dữ liệu mới. Điều chỉnh các siêu tham số như tốc độ học, số lượng epoch, kích thước batch để đảm bảo mô hình hội tụ tốt hơn.

Kết luận Chương 4

Kết thúc Chương 4, em đã trình bày được các kết quả lập trình và so sánh các mô hình với cỡ bước hằng, thuật toán quay lui. Kết quả cho độ chính xác dao động ở mức 72% - 76% ở hai mô hình. Với cỡ bước hằng thì cho độ chính xác nhỉnh hơn, nhưng không đáng kể. Tuy nhiên, với bộ dữ liệu có cỡ mẫu không quá lớn, nhưng mô hình hồi quy Softmax vẫn cho kết quả tương đối tốt. Hoàn thành được kết quả lập trình thực sự là một động lực để trong tương lai, em sẽ tìm hiểu thêm các mô hình để tối ưu kết quả và áp dụng trên nhiều bộ dữ liệu khác nhau.

Kết luận

Trong đồ án này, em đã tìm hiểu về hồi quy Softmax và ứng dụng trong bài toán phân loại chữ cái viết tay. Đồ án đã trình bày được cơ sở lý thuyết xây dựng hàm Softmax và cách hàm Softmax hoạt động, hàm mất mát entropy chéo, và các thuật toán tối ưu như thuật toán hướng giảm gradient với thuật toán quay lui.

Trong mô hình hồi quy Softmax để phân loại đa lớp, mỗi lớp sẽ có một bộ tham số của riêng lớp đó, một véc tơ $\mathbf{x}$ đại diện cho một điểm dữ liệu sẽ được nhân vô hướng với véc tơ trọng số của tất cả mọi lớp để thu được các giá trị điểm số, sau đó qua hàm Softmax, các giá trị điểm số này sẽ được chuyển về các giá trị xác suất và tổng các giá trị này bằng 1.

Để hàm Softmax trả về kết quả dự đoán tốt nhất, thì mục tiêu của chúng ta phải tìm bộ véc tơ tham số phù hợp nhất cho mô hình, muốn làm được điều này ta sẽ sử dụng một hàm mất mát với phụ thuộc vào các giá trị tham số này, sau đó thực hiện đạo hàm và tính gradient theo các tham số để tìm ra công thức tối ưu cho bộ tham số. Sau đó có thể dùng thuật toán hướng giảm gradient với cỡ bước hằng hoặc với thuật toán quay lui.

Đối với kết quả thực nghiệm trên bộ dữ liệu *letter* kết quả cho ra độ chính xác tương đối tốt. Kết quả với cỡ bước hằng cho ra độ chính xác trong khoảng 74% đến $\approx 76\%$, kết quả này cao hơn so với thuật toán quay lui khoảng 3%. Chúng ta có thể lý giải cho điều này vì có thế số vòng lặp trong cỡ bước hằng rất lớn, sai số nhỏ và thuật toán quay lui phải mất thêm một số bước giảm giá trị t_k. Ngoài ra, bộ dữ liệu cũng không quá lớn để mô hình học.

Việc chọn đề tài hồi quy Softmax đã giúp em mở rộng thêm kiến thức về học máy, trí tuệ nhân tạo, đặc biệt là với bài toán phân loại đa lớp, để từ đây trong tương lai em sẽ nghiên cứu sâu hơn về các phương pháp học máy khác, cũng như các mô hình toán học cho ra kết quả tối ưu tốt hơn trong quá trình huấn luyện mô hình. Đây cũng sẽ là nền tảng giúp em nghiên cứu sâu hơn về các mô hình phức tạp như học sâu, mạng nơ-ron, ứng dụng vào nhiều lĩnh vực khác nhau và mang lại nhiều giá trị.

Tài liệu tham khảo

[1] Vũ Hữu Tiệp, *Machine Learning cơ bản*, 2019.

[2] Nguyễn Thị Bạch Kim, *Giáo trình các phương pháp tối ưu: Lý thuyết và thuật toán*, NXB Bách khoa Hà Nội, 2008.

[3] https://bishnupadamajumder32.medium.com/confusion-matrix-5a1840998466

[4] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[5] https://www.bragitoff.com/wp-content/uploads/2021/12/Softmax-Activation.png