TRƯỜNG ĐẠI HỌC XÂY DỰNG HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

-----🙞🙜🕮🙞🙜-----

Icon

Description automatically generated

BÀI TẬP LỚN

**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

Đề tài:

**THUẬT TOÁN LOGISTIC REGRESSION**

**ỨNG DỤNG TRONG BÀI TOÁN PHÂN LOẠI**

**VỚI TẬP DỮ LIỆU MNIST**

Nhóm sinh viên thực hiện: Nhóm 1

Thành viên:

1.Nguyễn Minh Tùng Mssv: 218464

2.Lý Hải Yến Mssv: 1557164

3.Nguyễn Thị Toàn Mssv: 1549964

HÀ NỘI 06-2021

Mục Lục

[**I.** **Cơ sở dữ liệu MNIST của các chữ số viết tay** 4](#_Toc74343874)

[**II.** **Logictis Regression** 6](#_Toc74343875)

[**1)** **Giới thiệu** 6](#_Toc74343876)

[**2)** **Sigmoid function** 6](#_Toc74343877)

[**3)** **Hàm mất mát và phương pháp tối ưu** 7](#_Toc74343878)

[**III.** **Softmax Regression** 7](#_Toc74343879)

[**1)** **Giới thiệu** 7](#_Toc74343880)

[**2)** **Softmax function** 8](#_Toc74343881)

[**3)** **Hàm mất mát và phương pháp tối ưu** 9](#_Toc74343882)

[**3.1.** **Hàm mất mát** 9](#_Toc74343883)

[**3.2 Tối ưu hàm mất mát** 10](#_Toc74343884)

[**IV.** **Thuật toán Gradient Descent** 10](#_Toc74343885)

[**V.** **Kết quả thực nghiệm** 11](#_Toc74343886)

[**1)** **Part 1** 11](#_Toc74343887)

[**2)** **Part 2** 12](#_Toc74343888)

[**3)** **Part 3** 12](#_Toc74343889)

[**4)** **Part 4** 13](#_Toc74343890)

[**VI.** **KẾT LUẬN** 14](#_Toc74343891)

[**Tài liệu tham khảo** 15](#_Toc74343892)

**Lời nói đầu**

Đây là tài liệu để biểu diễn nội dung lý thuyết về thuật toán học máy softmax regression và trường hợp đặc biệt của nó là logistic regression để dự đoán cho bộ dữ liệu chữ viết MNIST.

Nhóm thực hiện bài tập lớn theo yêu cầu bài đưa ra để phân loại lớp chữ số cho cơ sở dữ liệu MNIST của các chữ số viết tay. Trong quá trình thực hiện bài tập lớn không tránh khỏi những sai sót, nhóm mong sẽ nhận được sự đánh giá và góp ý của thầy.

***Xin chân thành cảm ơn !***

1. **Cơ sở dữ liệu MNIST của chữ số viết tay**

Cơ sở dữ liệu MNIST của các chữ số viết tay có bộ huấn luyện gồm 60.000 mẫu và bộ kiểm tra gồm 10.000 mẫu. Nó là một tập hợp con của một tập hợp lớn hơn có sẵn từ NIST. Các chữ số đã được chuẩn hóa kích thước và căn giữa trong một hình ảnh có kích thước cố định.

Hình ảnh đen trắng (bilevel) hình ảnh ban đầu từ NIST đã được chuẩn hóa kích thước để vừa với hộp pixel 20x20 mà vẫn giữ nguyên tỷ lệ khung hình của chúng. Các hình ảnh thu được chứa các mức xám là kết quả của kỹ thuật khử răng cưa (anti-aliasing) được sử dụng bởi thuật toán chuẩn hóa. Các hình ảnh được căn giữa trong một hình ảnh 28x28 bằng cách tính toán khối lượng tâm của các điểm ảnh và dịch hình ảnh để định vị điểm này ở tâm của ảnh với kích thước 28x28.

Cơ sở dữ liệu MNIST được xây dựng từ Cơ sở dữ liệu đặc biệt 3 và Cơ sở dữ liệu đặc biệt 1 của NIST chứa các hình ảnh nhị phân của các chữ số viết tay. NIST ban đầu đã chỉ định SD-3 làm bộ huấn luyện của họ và SD-1 là bộ thử nghiệm của họ. Tuy nhiên, SD-3 sạch hơn và dễ nhận ra hơn nhiều so với SD-1. Lý do cho điều này có thể được tìm thấy trên thực tế là SD-3 được thu thập giữa các nhân viên của Cục Điều tra Dân số, trong khi SD-1 được thu thập trong số học sinh trung học. Việc rút ra kết luận hợp lý từ các thí nghiệm học tập đòi hỏi kết quả phải độc lập với việc lựa chọn tập huấn luyện và thử nghiệm trong số các tập mẫu hoàn chỉnh. Do đó, họ rút ra kết luận cần phải xây dựng một cơ sở dữ liệu mới bằng cách trộn các bộ dữ liệu của NIST.

Bộ đào tạo MNIST bao gồm 30.000 mẫu từ SD-3 và 30.000 mẫu từ SD-1. Bộ thử nghiệm của chúng tôi bao gồm 5.000 mẫu từ SD-3 và 5.000 mẫu từ SD-1. Bộ đào tạo 60.000 mẫu bao gồm các ví dụ từ khoảng 250 nhà văn. Họ đảm bảo rằng tập hợp các tác giả của tập huấn luyện và tập thử nghiệm là rời rạc, không liên quan đến nhau.

SD-1 chứa 58.527 hình ảnh chữ số được viết bởi 500 nhà văn khác nhau, dữ liệu trong SD-1 bị xáo trộn. Ngược lại với SD-3, trong đó các khối dữ liệu từ mỗi người ghi xuất hiện theo thứ tự. Danh tính người viết cho SD-1 hiện có sẵn và họ đã sử dụng thông tin này để sắp xếp người viết. Sau đó, họ chia SD-1 thành hai: các ký tự được viết bởi 250 nhà văn đầu tiên được đưa vào bộ huấn luyện mới của chúng tôi. 250 nhà văn còn lại đã được đặt trong bộ thử nghiệm của họ.

Vì vậy, chúng tôi có hai bộ với gần 30.000 mẫu mỗi bộ. Tập huấn luyện mới đã được hoàn thành với đủ các mẫu từ SD-3, bắt đầu từ mẫu # 0, để tạo thành một tập hợp đầy đủ gồm 60.000 mẫu huấn luyện. Tương tự, bộ thử nghiệm mới đã được hoàn thành với các ví dụ SD-3 bắt đầu từ mẫu # 35.000 để tạo thành một tập hợp đầy đủ với 60.000 mẫu thử nghiệm. Và chỉ có một tập hợp con gồm 10.000 hình ảnh thử nghiệm (5.000 từ SD-1 và 5.000 từ SD-3) có sẵn trên trang web này. Có sẵn bộ huấn luyện mẫu đầy đủ 60.000.

SD-3-Special Database 3(Cơ sở dữ liệu đặc biệt 3)

SD-1-Special Database 1(Cơ sở dữ liệu đặc biệt 1)

Dữ liệu có sẵn trên trang web <http://yann.lecun.com/exdb/mnist/> . Dữ liệu gồm bốn file:

* train-images-idx3-ubyte : tập huấn luyện với dữ liệu là hình ảnh
* train-labels-idx1-ubyte : tập huấn luyện với dữ liệu là các nhãn
* t10k-images-idx3-ubyte : tập thử nghiệm với dữ liệu là hình ảnh
* t10k-labels-idx1-ubyte : tập thử nghiệm với dữ liệu là nhãn

Dữ liệu được lưu trữ ở định dạng tệp rất đơn giản được thiết kế để lưu trữ vectơ và ma trận nhiều chiều. Tất cả các số nguyên trong tệp được lưu trữ ở định dạng MSB first (high endian). Các điểm ảnh được sắp xếp theo hàng. Giá trị pixel là 0 đến 255. 0 có nghĩa là nền sau (trắng), 255 có nghĩa là nền trước (đen). Giá trị các nhãn từ 0-9.

train-labels-idx1-ubyte:

|  |  |  |  |
| --- | --- | --- | --- |
| [offset] | [type] | [value] | [description] |
| 0000 | 32 bit integer | 0x00000801(2049) | magic number (MSB first) |
| 0004 | 32 bit integer | 60000 | số lượng mẫu |
| 0008 | unsigned byte | ?? | Nhãn |
| 0009 | unsigned byte | ?? | Nhãn |
| …… |  |  |  |
| xxxx | unsigned byte | ?? | Nhãn |

train-images-idx3-ubyte:

|  |  |  |  |
| --- | --- | --- | --- |
| [offset] | [type] | [value] | [description] |
| 0000 | 32 bit integer | 0x00000803(2051) | magic number (MSB first) |
| 0004 | 32 bit integer | 60000 | số lượng ảnh |
| 0008 | 32 bit integer | 28 | Số cột |
| 0012 | 32 bit integer | 28 | Số hàng |
| 0016 | unsigned byte | ?? | pixel |
| 0017 | unsigned byte | ?? | pixel |
| …… |  |  |  |
| xxxx | unsigned byte | ?? | pixel |

Tương tự với 2 tập còn lại.

***Định dạng tập tin IDX:***

Magic number là một số nguyên (MSB first), 2 byte đầu tiên luôn là 0.

Kích thước trong chiều số 0

Kích thước trong chiều số 1

Kích thước trong chiều số 2

….

Kích thước trong chiều số N

Byte thứ ba mã kiểu dữ liệu:

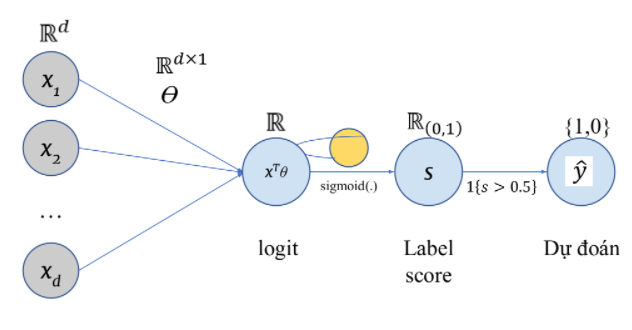
* 0x08: unsigned byte
* 0x09: signed byte
* 0x0B: short (2 bytes)
* 0x0C: int (4 bytes)
* 0x0D: float (4 bytes)
* 0x0E: double (8 bytes)

Byte thứ 4 mã số kích thước của vectơ/ma trận: 1 đối với vectơ, 2 đối với ma trận. Kích thước trong mỗi chiều là số nguyên 4 byte (MSB first, high endian, giống như trong hầu hết các bộ xử lý không phải của Intel).

1. **Logictis Regression**
2. **Giới thiệu**

Logistic regression là 1 thuật toán phân loại nhị phân được dùng để phân các đối tượng vào 2 lớp thường được kí hiệu là 0 và 1. Đầu ra dự đoán của logistic regression thường được viết chung dưới dạng:

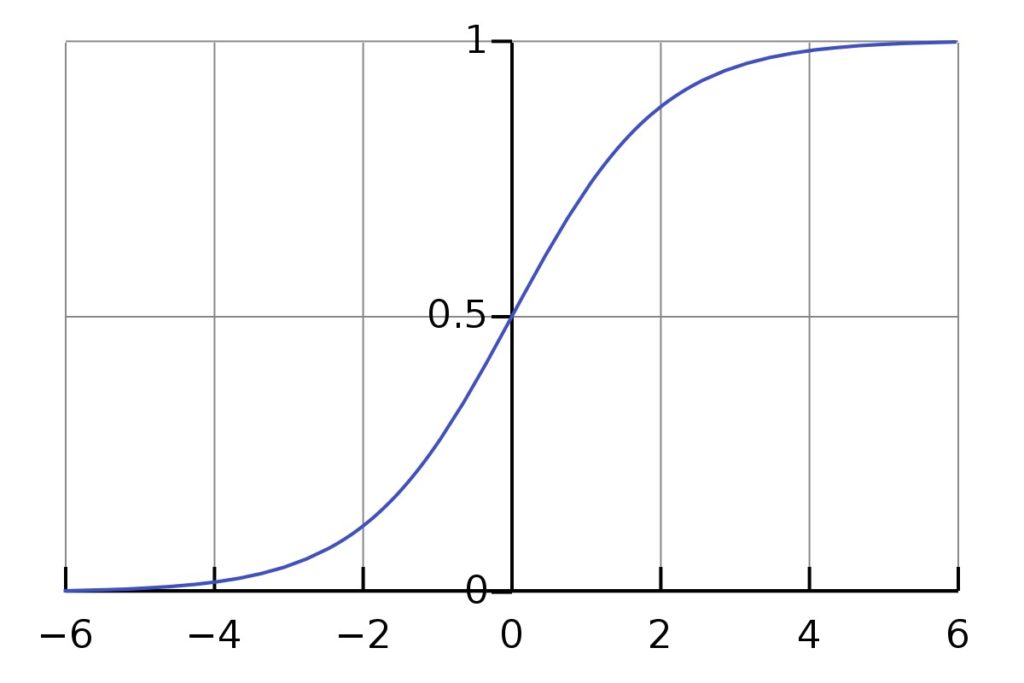
Trong đó g là hàm sigmoid.



1. **Sigmoid function**

Sigmoid Function (Hàm Sigmoid) còn được gọi là đường cong Sigmoid là một hàm toán học có đặc trưng là đường cong hình chữ S. Giá trị của hàm sigmoid bị chặn trong khoảng (0,1). Có công thức là:

|  |  |
| --- | --- |
|  | (1) |



Gọi ước lượng P(y|x) là phân bố xác suất của label y khi biết x ta có:

|  |  |
| --- | --- |
|  | (2) |

|  |  |
| --- | --- |
|  | (3) |

1. **Hàm mất mát và phương pháp tối ưu**

Hàm mất mát cần cực tiểu hóa trong mô hình Logistic regression như sau:

|  |  |
| --- | --- |
|  | (4) |

Đạo hàm của hàm mất mát khi có hiệu chỉnh L2 với từng phần tử trong θ:

|  |  |  |
| --- | --- | --- |
|  | với j=0 | (5) |

|  |  |  |
| --- | --- | --- |
|  | với j | (6) |

1. **Softmax Regression**
2. **Giới thiệu**

Các bài toán phân lớp thực tế thường có rất nhiều lớp dữ liệu, các bộ phân lớp nhị phân tuy có thể kết hợp được với nhau để giải quyết bài toán, nhưng chúng vẫn có những hạn chế nhất định. Trong phần này, ta đề cập đến Softmax Regression (Multinomial Logistic, Multi-class Logistic Regression) là một phương pháp mở rộng của Logistic Regression mà chúng ta có thể sử dụng để phân đa lớp. Nó cũng chính là một trong những thành phần phổ biến nhất trong các bộ phân lớp hiện nay.

Với bài toán phân lớp nhị phân sử dụng logistic regression, đầu ra là một số thực trong khoảng (0,1), đóng vai trò như là xác suất để đầu vào thuộc một trong hai lớp 0 hoặc 1. Ý tưởng này cũng có thể mở rộng cho bài toán phân đa lớp (10 lớp với ví dụ phân loại bộ chữ số viết tay Mnist), ở đó có đầu ra là C các giá trị, mỗi giá trị đóng vai trò như xác suất để đầu vào rơi vào lớp tương ứng. Như vậy, các đầu ra này liên kết với nhau qua việc chúng đều là số dương và có tổng bằng 1.

1. **Softmax function**

Chúng ta cần một mô hình xác suất sao cho với mỗi input x, thể hiện xác suất để input đó rơi vào lớp i. Vậy điều kiện cần là các ti phải dương và tổng của chúng bằng 1, ngoài ra giá trị zi= càng lớn thì xác suất dữ liệu rơi vào lớp thứ i càng cao. Điều kiện cuối cùng này chỉ ra rằng ta cần một quan hệ đồng biến ở đây.

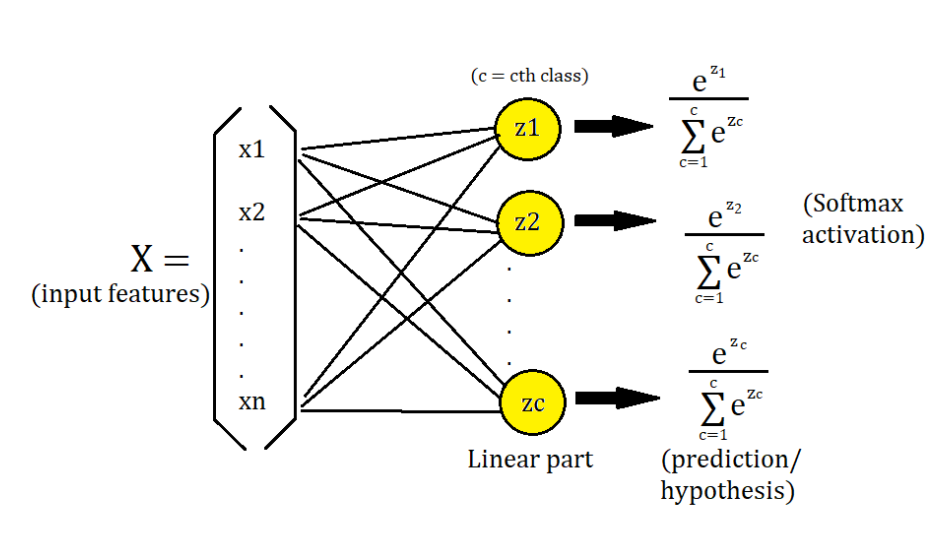
Chú ý rằng zi có thể nhận giá trị cả âm và dương vì nó là một tổ hợp tuyến tính của các thành phần của vector đặc trưng x. Một hàm số khả vi đơn giản có thể chắc chắn biến zi thành một giá trị dương và hơn nữa, đồng biến, là hàm exp(zi) = . Điều kiện khả vi để thuận lợi cho việc sử dụng đạo hàm cho việc tối ưu. Điều kiện cuối cùng, tổng các bằng 1 có thể được đảm bảo nếu

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

Mối quan hệ này, với mỗi phụ thuộc vào tất cả các zi, thỏa mãn tất cả các điều kiện đã xét: dương, tổng bằng 1, giữ được thứ tự của zi. Hàm số này được gọi là softmax function. Lúc này, ta có thể coi rằng

|  |  |
| --- | --- |
|  | (8) |

Trong đó, được hiểu là xác suất để một điểm dữ liệu x rơi vào lớp thứ i nếu biết tham số mô hình là ma trận trọng số W



với vector đầu vào z có độ dài tương đương với số lớp c

|  |  |
| --- | --- |
|  | (9) |

(với W là vector trọng số, X là vector feature của một mẫu huấn luyện và b là vector hệ số bias).

1. **Hàm mất mát và phương pháp tối ưu**
   1. **Hàm mất mát**

Hàm mất mát của softmax regression được xây dựng dựa trên bài toán tối thiểu sự khác nhau giữa đầu ra dự đoán và đầu ra thự sự y ( ở dạng one-hot). Khi cả hai là các vector thể hiện xác suất, khoảng cách giữa chúng thường được đo bằng một đại lượng được gọi là cross entropy. Một điểm nổi bật của đại lượng này là giá trị của nó đạt giá trị nhỏ nhất khi hai vector xác suất bằng nhau, và rất lớn khi hai vector đó lệch nhau nhiều.

Cross entropy giữa hai vector phân phối p và q rời rạc được định nghĩa bởi

|  |  |
| --- | --- |
|  | (10) |

Trong trường hợp có c lớp dữ liệu, mất mát giữa đầu ra dự đoán và đầu ra thực sự của một điểm dữ liệu xi với label (one-hot) yi được tính bởi

|  |  |
| --- | --- |
|  | (11) |

Kết hợp các điểm dữ liệu, hàm mất mát cho softmax regression được xác định bởi

|  |  |
| --- | --- |
|  | (12) |

Ta có thể thêm hiệu chỉnh L2 để tránh overfitting,

|  |  |
| --- | --- |
|  | (13) |

Trong đó trong đó, F là kí hiệu Frobenius norm(cũng là Euclidean norm) căn bậc 2 của tổng các bình phương của các w:

|  |  |
| --- | --- |
|  | (14) |

### **3.2 Tối ưu hàm mất mát**

Đạo hàm của hàm mất mát (không có hiệu chỉnh) đối với một điểm dữ liệu cụ thể liên quan đến các tham số W:

|  |  |
| --- | --- |
|  | (15) |

Đạo hàm của hàm mất mát có hiệu chỉnh trên một điểm dữ liệu:

|  |  |
| --- | --- |
|  | (16) |

Đạo hàm của hàm mất mát có hiệu chỉnh trên toàn tập dữ liệu:

|  |  |
| --- | --- |
|  | (17) |

Dùng công thức sai phân thuận để kiểm tra độ chính xác của giá trị đạo hàm theo công thức (16):

|  |  |
| --- | --- |
|  | (18) |

1. **Thuật toán Gradient Descent**

Hướng tiếp cận phổ biến nhất là xuất phát từ một điểm mà chúng ta coi là gần với nghiệm của bài toán, sau đó dùng một phép toán lặp để tiến dần đến điểm cần tìm, tức đến khi đạo hàm gần với 0. Gradient Descent (viết gọn là GD) là một trong những phương pháp được dùng nhiều nhất.

Với bài tập này chúng ta cần tìm global minimum cho hàm Loss. Quy tắc phải nhớ là luôn luôn đi ngược hướng với đạo hàm. Bắt đầu bằng một điểm dự đoán θ0 hay còn được kí hiệu là W,sau đó, ở vòng lặp thứ t, quy tắc cập nhật là:

* Khởi tạo θj:

: learning rate

Learning rate:

1. Với learning rate nhỏ , tốc độ hội tụ rất chậm. Trong thực tế, khi việc tính toán trở nên phức tạp, learning rate quá thấp sẽ ảnh hưởng tới tốc độ của thuật toán rất nhiều, thậm chí không bao giờ tới được đích.
2. Với learning rate lớn , thuật toán tiến rất nhanh tới gần đích sau vài vòng lặp. Tuy nhiên, thuật toán không hội tụ được vì bước nhảy quá lớn, khiến nó cứ quẩn quanh ở đích.

Việc lựa chọn learning rate rất quan trọng trong các bài toán thực tế. Việc lựa chọn giá trị này phụ thuộc nhiều vào từng bài toán và phải làm một vài thí nghiệm để chọn ra giá trị tốt nhất.

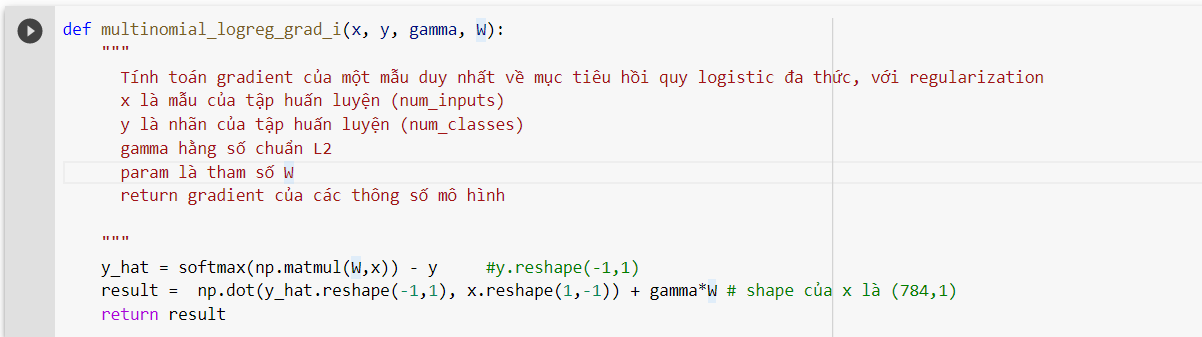
Tính chất:

* Với và thì tăng
* Với và thì giảm

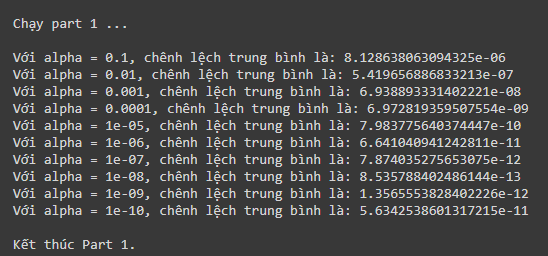
1. **Kết quả thực nghiệm**
2. **Part 1**

Implement hàm multinomial\_logreg\_loss\_i tính toán hàm loss cho 1 mẫu cụ thể trong tập dữ liệu.

Implement hàm multinomial\_logreg\_grad\_i để tính toán hàm loss được đạo hàm cho 1 mẫu cụ thể trong tập dữ liệu. Sử dụng hàm softmax để tính label dự đoán, sau đó lấy kết quả ta vừa tính được trừ đi label thực tế, sau đó lấy kết quả y\_hat là sự khác biệt giữa label dự đoán và label thực tế nhân với x.T là 1 mẫu dữ liệu được chuyển vị. Để tính đạo hàm của hàm mất mát, theo công thức (16) có sử dụng chuẩn L2.



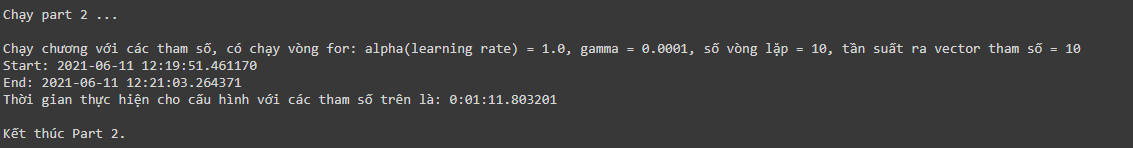
Output sau khi chạy hàm test\_gradient:



Kết luận: chệnh lệnh giữa đạo hàm theo công thức (16) với công thức (18) là không lớn, cho thấy chúng ta có thể tin tưởng vào công thức đạo hàm hàm mất mát (16) để sử dụng cho thuật toán GD. Và có giá trị nhỏ nhất từ alpha 10-9 đến 10-11.

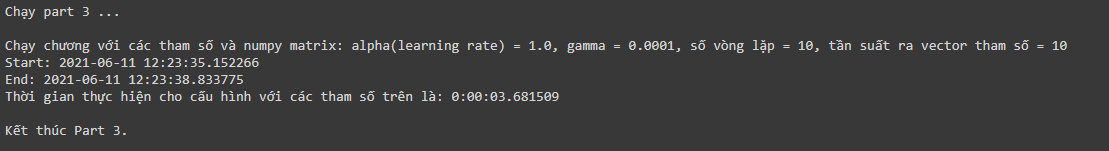
1. **Part 2**

Để chạy thuật toán Gradient Descent với 10 vòng lặp với các tham số giả thiết, cần khoảng 80s:



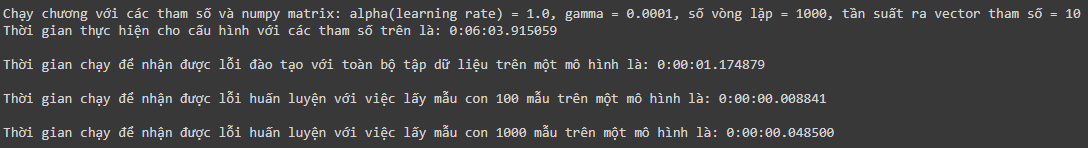
Với số vòng lặp là 1000, ta mất khoảng 8000s (hơn 2 tiếng) để hoàn thành thuật toán.

1. **Part 3**



Khi sử dụng Numpy matrix ta thấy thời gian chạy nhanh hơn khi sử dụng vòng for trong part 2. Vì vòng for duyệt qua từng phần tử trong ma trận tập huấn luyện, numpy matrix duyệt toàn bộ ma trận rồi mới thực hiện phép nhân.

1. **Part 4**



Cần phải có kích thước tập mẫu con là nhỏ như là 100 sẽ cho thấy rõ lỗi của mô hình hơn là tập con với 1000 mẫu.

Phương pháp tính lỗi trên toàn bộ tập dữ liệu sẽ lâu hơn phương pháp ước tính lỗi ở tập con với 100 và 1000 mẫu vì nó phải tính lỗi trên nhiều mẫu hơn.

1. **KẾT LUẬN**

Thông qua việc tìm hiểu và nghiên cứu đề tài này giúp nhóm có cái nhìn toàn diện hơn trong việc ứng dụng trí tuệ nhân tạo vào giải quyết vấn đề thực tế. Đây là bài toán cổ điển trong trí tuệ nhân tạo cho các thuật toán phân lớp. Do thời gian có hạn nên không tránh khỏi những sai sót, mong thầy góp ý, đánh giá giúp nhóm hoàn thiện hơn.

# **Tài liệu tham khảo**

* Cơ sở dữ liệu về chữ viết tay MNIST

<http://yann.lecun.com/exdb/mnist/>

* Thuật toán Gradient Descent

<https://machinelearningcoban.com/2017/01/12/gradientdescent/>

* Machine Learning cơ bản (Vũ Hữu Tiệp)
* Bài giảng nhập môn trí tuệ nhân tạo(Nguyễn Đình Quý)

*Hà Nội, ngày 11 tháng 6 năm 2021*

Tác giả BTL

Nhóm 1