**TRƯỜNG ĐẠI HỌC XÂY DỰNG**

**KHOA CÔNG NGHỆ THÔNG TIN  
------------o0o------------**

****

**BÁO CÁO BÀI TẬP LỚN**

***MÔN:***

**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

***GVHD: Nguyễn Đình Quý.***

**Nhóm sinh viên thực hiện***:*

Nguyễn Trần Lê Tuấn 64CS3 MSSV 1553564

Vương Trung Thành 64CS3 MSSV 186864

Dương Gia Khánh 64CS3 MSSV 1526864

***Hà Nội, ngày 7 tháng 6 năm 2021***

**NỘI DUNG**

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT LOGISTIC REGRESSION 3](#_Toc74046060)

[**1.1. Định nghĩa** 3](#_Toc74046061)

[**1.2.** **Ước lượng xác suất** 3](#_Toc74046062)

[**1.3.** **Xây dựng hàm mất mát (loss function)** 5](#_Toc74046063)

[**1.4.** **Ước lượng tham số bằng Gradient Descent** 6](#_Toc74046064)

[CHƯƠNG 2. SOFTMAX REGRESSION 6](#_Toc74046065)

[**2.1. Giới thiệu về mô hình và ước lượng xác suất** 6](#_Toc74046066)

[**2.2. Xây dựng hàm mất mát (Loss Function)** 7](#_Toc74046067)

[2.2.1. One hot coding 7](#_Toc74046068)

[2.2.2. Cross Entropy 8](#_Toc74046069)

[2.2.3. Hàm mất mát cho Softmax Regression: 8](#_Toc74046070)

[CHƯƠNG 3. BÁO CÁO SỐ LIỆU THỰC NGHIỆM 9](#_Toc74046071)

[**Part 1.The gradient of the loss function** 9](#_Toc74046072)

[**Part 2. Gradient Descent:** 11](#_Toc74046073)

[**Part 3.Tối ưu trong tính toán dữ liệu** 13](#_Toc74046074)

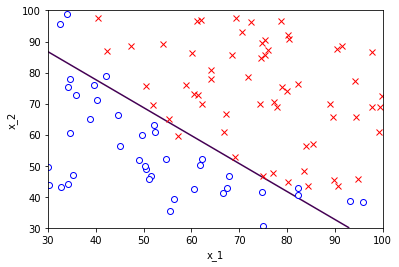
[**Part 4. Evaluating gradient descent** 15](#_Toc74046075)

# **CHƯƠNG 1. CƠ SỞ LÝ THUYẾT LOGISTIC REGRESSION**

## **1.1. Định nghĩa**

Phương pháp hồi quy logistic là một mô hình hồi quy nhằm dự đoán giá trị đầu ra **rời rạc (discrete target variable)  *y* ứng với một véc-tơ đầu vào x.** Việc này tương đương với chuyện phân loại các đầu vào **x** vào các nhóm *y* tương ứng.  **Logistic Regression (hay còn gọi là Logit Regression)** được sử dụng phổ biến để ước lượng xác suất 1 điểm dữ liệu có thể thuộc về 1 lớp nào đó (ví dụ tính xác suất để 1 email là spam). Nếu xác suất > 50% thì mô hình dự đoán có khả năng cao là điểm dữ liệu đó thuộc về lớp 1 (nhãn là 1) hoặc ngược lại (nhãn là 0). **Việc này được gọi là phân loại nhị phân (chỉ có 0 với 1)**

Ví dụ, xem một bức ảnh có chứa một con mèo hay không. Thì ở đây ta coi đầu ra *y*=1 nếu bức ảnh có một con mèo và *y*=0 nếu bức ảnh không có con mèo nào. Đầu vào **x** ở đây sẽ là các pixel một bức ảnh đầu vào.



Mô hình phân loại nhị phân

**Để đơn giản, trước tiên ta sẽ cùng đi tìm hiểu mô hình và cách giải quyết cho bài toán phân loại nhị phân tức là  *y*={0,1}. Sau đó ta mở rộng cho trường hợp nhiều nhóm sau****.**

## **Ước lượng xác suất**

Mục tiêu của bài toán là phân loại xem giá trị đầu vào từ tập dữ liệu đưa ra kết quả dữ liệu đó thuộc lớp 0 hay lớp 1. Để thực hiện được điều đó, trước tiên ta cần thiết lập một hàm logistic để đánh giá khả năng thuộc vào một trong hai lớp của dữ liệu đó. Sử dụng phương pháp thống kê ta có thể coi rằng khả năng một đầu vào **x** nằm vào một nhóm *y*​ là xác suất nhóm *y*​ khi biết **x**: **P** (*y* | **x**).

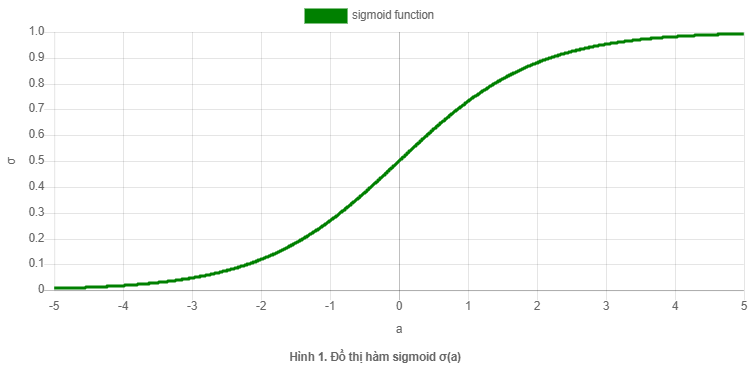
**P** (*y* | **x**) = hW (**x**) = =

Trong đó:

+ **x** là thuộc tính đầu vào (có cộng thêm bias), **w** là trọng số tương ứng.

+ Hàm được gọi là hàm **sigmoid**, đầu ra sẽ là khoảng giá trị từ 0 đến 1.

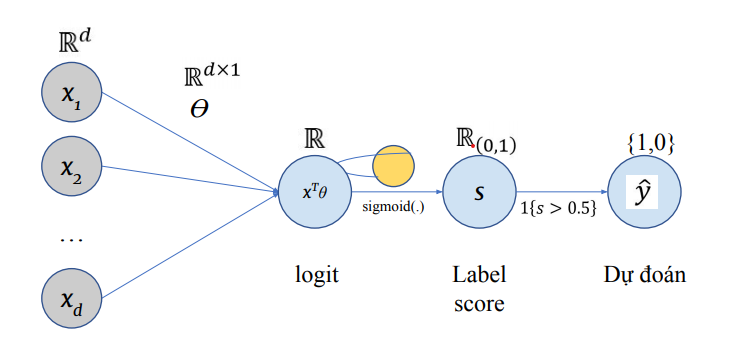
=



Sau khi đã tính được xác suất **P** (*y* | **x**) = hW (**x**) ta sẽ dễ dàng xác định được **x** thuộc về lớp nào, hay *y:*

Để ý rằng  < 0.5 khi t < 0 và  0.5 khi t ≥ 0. Vì vậy hồi quy Logistic sẽ dự đoán là 1 khi  là dương và 0 khi  âm.

**Dưới đây là mô hình về phân lớp nhị phân bằng logistic regression dưới dạng biểu diễn đơn giản của một Multiple Perceptron (Mạng nơ ron truyền thẳng)**

****

## **Xây dựng hàm mất mát (loss function)**

Mục tiêu của việc training sẽ là tìm ra vector tham số **w** để mô hình ước lượng xác suất cao cho các điểm dữ liệu thuộc vào lớp 1 và xác suất thấp cho các điểm dữ liệu thuộc vào lớp 0

**Để giải quyết bài toán phân loại này ta lại sử dụng phương pháp đánh giá MLE (Maximum Likelihood Estimation) bằng 2 bước:**

* Biểu diễn hàm tối ưu dạng hàm log-likelihood của \theta*θ*
* Tìm **w** để hàm này đạt cực đại

Với giả thuyết ta chỉ có 2 nhãn *y* =0 hoặc *y*=1, ta có thể sử dụng **hàm phân phối Bernoulli distribution**  để ước lượng xác suất của mỗi nhãn như sau:

**P**(***Y=y |X=x***) =

Trong đó: *p*   Từ đây ta có tích xác suất hợp của cả tập dữ liệu như sau:

Trực tiếp tối ưu hàm số này theo **w** nhìn qua không đơn giản! Hơn nữa, khi **m** lớn, tích của **m**số nhỏ hơn 1 có thể dẫn tới **sai số trong tính toán (numerial error)** vì tích là một số quá nhỏ. Một phương pháp thường được sử dụng đó là **lấy logarit tự nhiên (cơ số *e*) của likelihood function**biến phép nhân thành phép cộng và để tránh việc số quá nhỏ. Sau đó lấy ngược dấu để được một hàm và coi nó là hàm mất mát. Lúc này bài toán tìm giá trị lớn nhất (maximum likelihood) trở thành bài toán tìm giá trị nhỏ nhất của hàm mất mát (hàm này còn được gọi là negative log likelihood):

*J(***w***)* =

Trong đó, *m* là kích cỡ của tập dữ liệu,  lớp tương ứng của dữ liệu thứ *i* trong tập dữ liệu, là xác suất tương ứng khi tính với mô hình cho dữ liệu thứ *i*.

## **Ước lượng tham số bằng Gradient Descent**

Để tối ưu hàm *J(***w***)* trên, ta lại sử dụng các phương pháp Gradient Descent để thực hiện. Ở đây, đạo hàm của hàm log trên có thể được tính như sau:

 ta sẽ cập nhập tham số sau mỗi vòng lặp như sau và biểu diễn dưới dạng vector hóa:

**w\_new = w\_old**

# **CHƯƠNG 2. SOFTMAX REGRESSION**

## **2.1. Giới thiệu về mô hình và ước lượng xác suất**

Trong dự án nhận dạng chữ số viết tay với tập dữ liệu MNIST có sẵn lần này, chúng tôi sẽ trình bày quy trình thực thi và đánh giá Gradient Descent cho thực nghiệm sao cho giảm thiểu các rủi ro trên mô hình hồi quy Logistic đa lớp (với các lớp từ 0 cho tới 9). Chúng ta sẽ bắt đầu bằng cách đánh giá mô hình hồi quy Logistic đa lớp như một phương thức. Xét một phân loại đa lớp với *k* lớp trong đó các dự đoán *ŷ* phân bố trên *k* lớp.

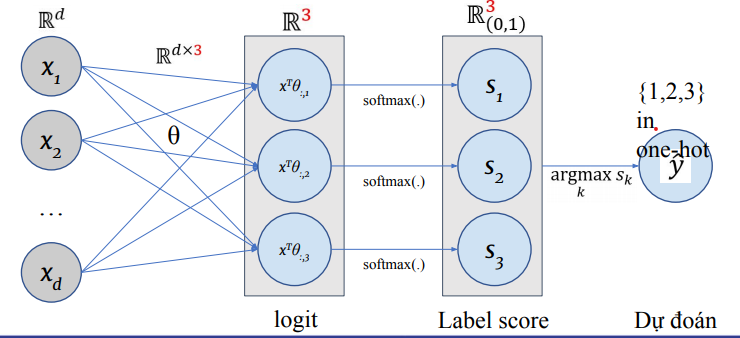
*ŷ* ∈ , *ŷi ≥* 0 với ∀ i ∈ {1, … , k} và

Trong ví dụ nhãn *y* là vector one-hot xác định một lớp đơn. , mỗi *yi* {0,1}, và tồn tại một *i* {1, …, k} sao cho *yi* =1. Trong ví dụ bản thân chúng là các vector . Hồi quy Logistic đa lớp sử dụng một mô hình tuyến tính giả thuyết với một hàm softmax. Đặc biệt, truyền vào một ma trận tham số , được hàm giả thuyết:

(**x**) = softmax ()

Hàm softmax nối từ đến và được định nghĩa bằng:

*Sk = P(y(i) = k* | ***x****(i); ) =*

**

**Hàm Softmax giúp ta chuẩn hóa các giá trị sao cho các label score *Sk* tạo thành một phân phối xác suất (tổng các giá trị bằng 1).**

## **2.2. Xây dựng hàm mất mát (Loss Function)**

### **2.2.1. One hot coding**

**Với cách biểu diễn mạng neuron truyền thẳng như trên,** mỗi output sẽ không còn là một giá trị tương ứng với mỗi class nữa mà sẽ là một vector có đúng 1 phần tử bằng 1, các phần tử còn lại bằng 0. Phần tử bằng 1 nằm ở vị trí tương ứng với class đó, thể hiện rằng điểm dữ liệu đang xét rơi vào class này với xác suất bằng 1.

Khi sử dụng mô hình Softmax Regression, với mỗi đầu vào **x**, ta sẽ có đầu ra dự đoán là *ŷ =* softmax(**x**).Trong khi đó, đầu ra thực sự chúng ta có là vector *y* được biểu diễn dưới dạng one-hot coding.

### **2.2.2. Cross Entropy**

Cross Entropy giữa hai phân phối *ŷ* và *y:*

L(*ŷ ,y*) =,

Trong Logistic Regression, chúng ta cũng có hai phân phối đơn giản. (i) Đầu ra thực sự của điểm dữ liệu đầu vào **xi** có phân phối xác suất là [*yi*;1− *yi*]với *yi* là xác suất để điểm dữ liệu đầu vào rơi vào class thứ nhất (bằng 1 nếu *yi*= 1, bằng 0 nếu *yi*= 1,). (ii). Đầu ra dự đoán của điểm dữ liệu đó là =sigmoid(**x**) là xác suất để điểm đó rơi vào class thứ nhất. Xác suất để điểm đó rơi vào class thứ hai có thể được dễ dàng suy ra là 1-. Vì vậy, hàm mất mát trong Logistic Regression:

*J(***w***)* =

chính là một trường hợp đặc biệt của Cross Entropy.

**Với Softmax Regression, trong trường hợp có *k* classes, loss giữa đầu ra dự đoán và đầu ra thực sự của một điểm dữ liệu xi được tính bằng:**

*J*(**W;** xi; yi)=,

### **2.2.3. Hàm mất mát cho Softmax Regression:**

*J(***w**) =

=

Nhắc lại 1 chút, như vậy hàm chi phí của hồi quy logistic có thể viết dưới dạng:

*J(***w***)* =

=

Lúc này hàm chi phí của Logistic regression nhìn khá giống với hàm chi phí của Softmax regression, chỉ khác là chúng ta tính tổng các xác suất của K lớp khác nhau. Như vậy:

=

**Bằng cách đạo hàm *J(*w) ta sẽ tìm được gradient như sau:**

=

là một vector có phần tử thứ j là là đạo hàm riêng của  đối với phần tử thứ j của

Ta sẽ cập nhật tham số sau mỗi vòng lặp như sau và lấy vector hóa:

**w\_new = w\_old**

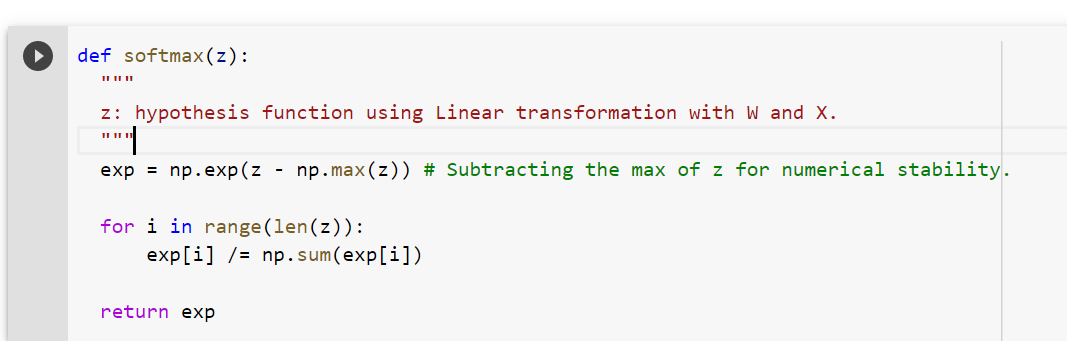
# 

# **CHƯƠNG 3. BÁO CÁO SỐ LIỆU THỰC NGHIỆM**

## **Part 1.The gradient of the loss function**

Trong chương này chúng tôi sẽ tiến hành thực hiện mô hình theo lí thuyết đã nêu ở trên, kết hợp sử dụng ngôn ngữ Python là công cụ tính toán.Trước tiên chúng tôi tính hàm softmax nhằm mục đích chuyển các giá trị từ hàm giá thiết về phân phối có tổng xác suất các giá trị là 1.

(x) = softmax () =



Tiếp theo, chúng tôi dựa vào hàm mất mát cho mô hình Softmax Regression:

*J(***w**) =

=

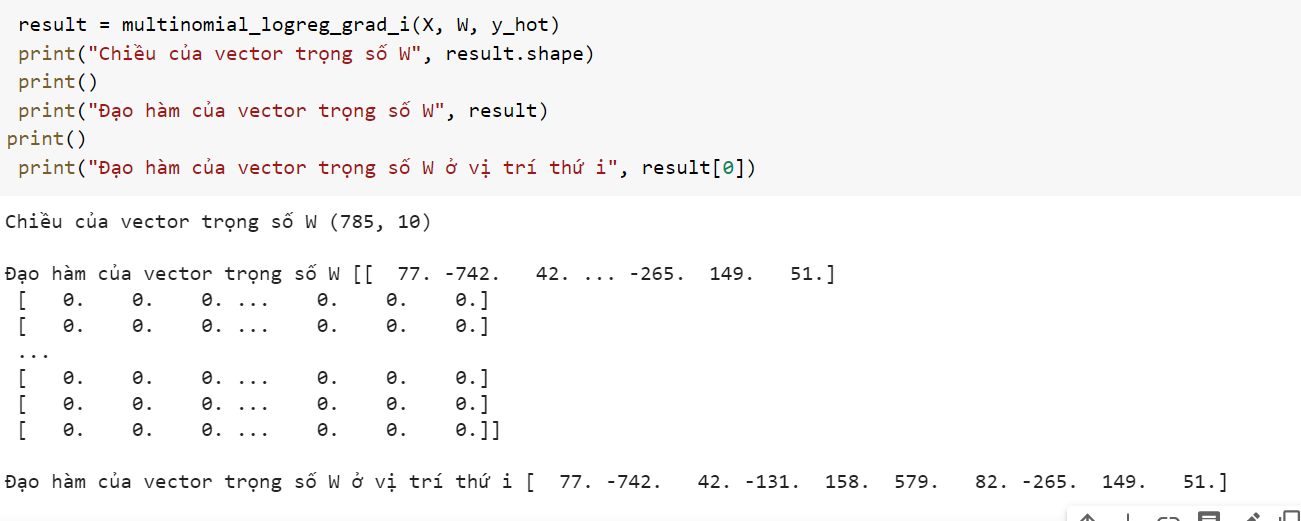
Lấy đạo hàm của hàm mất mát:

=

Kết quả cuối cùng sau khi Vector hóa:

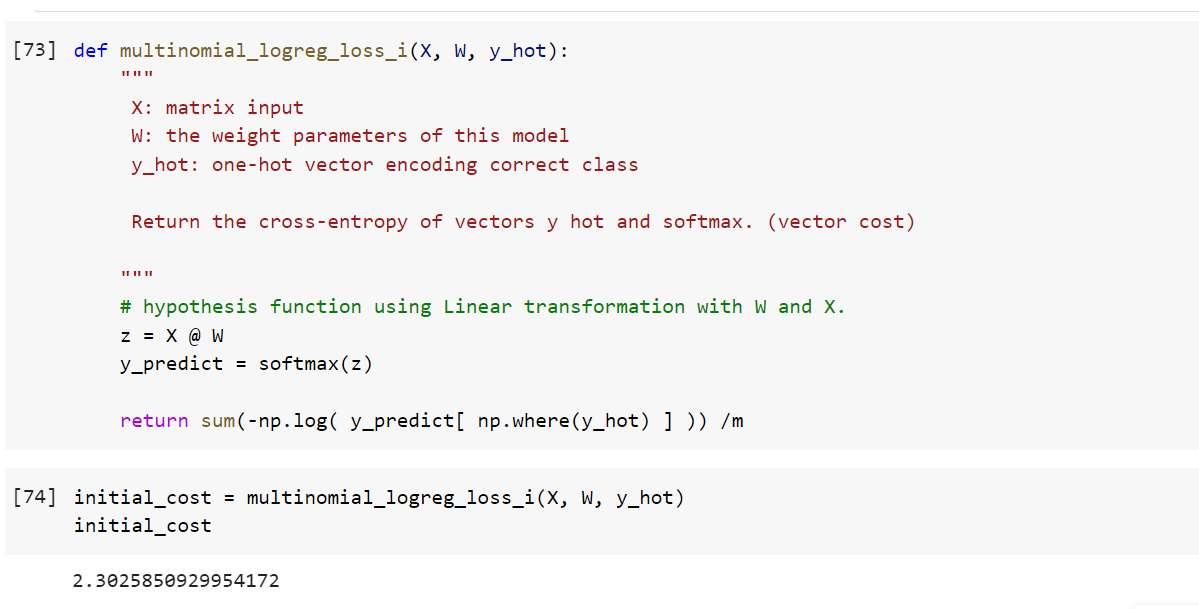
= (softmax () – y)

Dựa vào công thức trên, ta tạo hàm **multinomial\_logred\_grad\_i** để tính gradient cho một mẫu (x, y) với tham số W. Với kết quả thu được trong vòng lặp đầu tiên như sau:



**Dựa vào công thức cross-entropy cho một điểm dữ liệu**

L(*ŷ , y*) =,

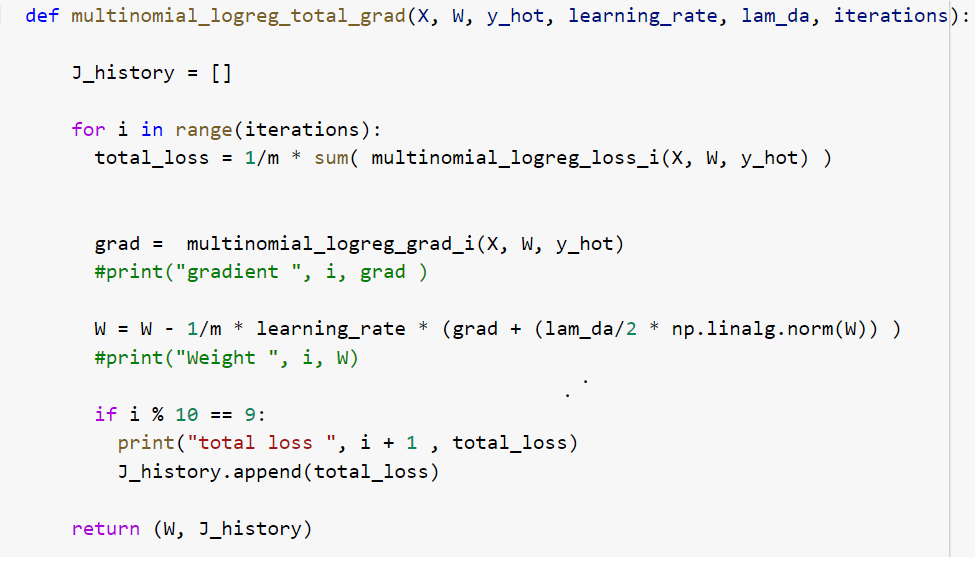
viết được hàm **multinomial\_logreg\_loss\_i** để tính sai số cho một mẫu (x, y) với tham số W

## **Part 2. Gradient Descent**

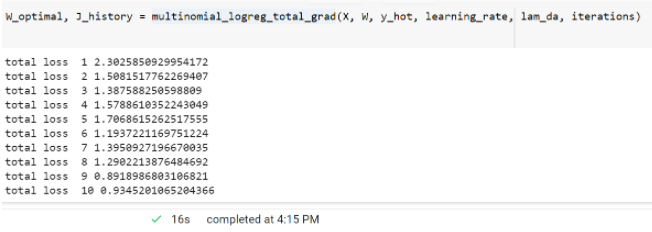
Thuật toán Gradient Descent được áp dụng như sau:

**Ta sẽ sử dụng vòng lặp cho đến khi trọng số theta hội tụ và có nghiệm tương đối tối ưu là global minimum (cực tiểu toàn cục)**

Để triển khai thuật toán Gradient Descent, dùng hàm **multinomial\_logreg\_total\_grad** để tính sai số cho toàn tập dữ liệu

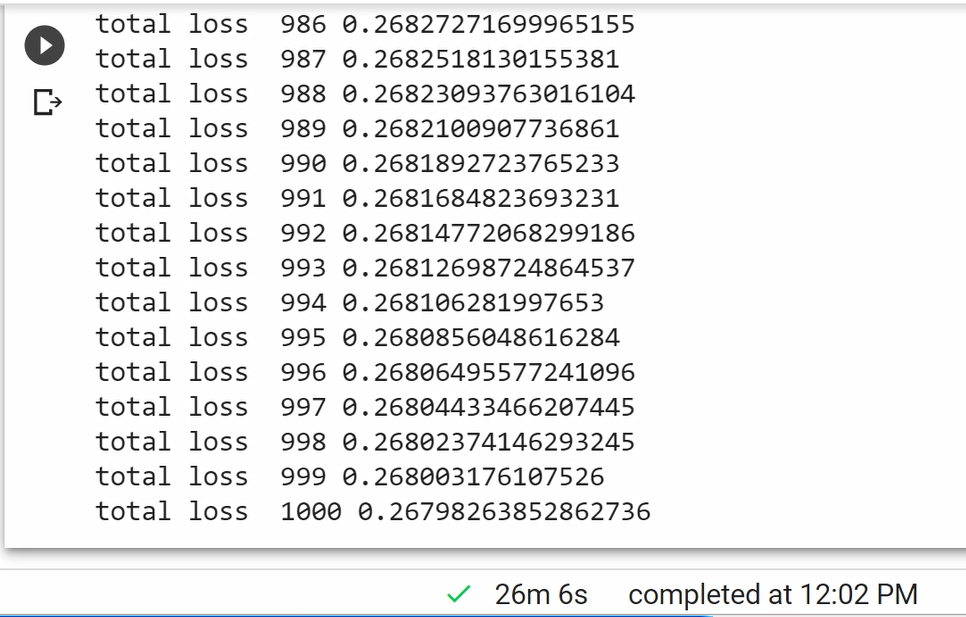


**Ta truyền tham số chuẩn hóa L2 regularization là lamda\_da = 0.0001 (Việc chuẩn hóa này để giúp mô hình tránh khởi việc Overfitting), learning\_rate = 1.0, iterations = 10. Được kết quả như sau:**



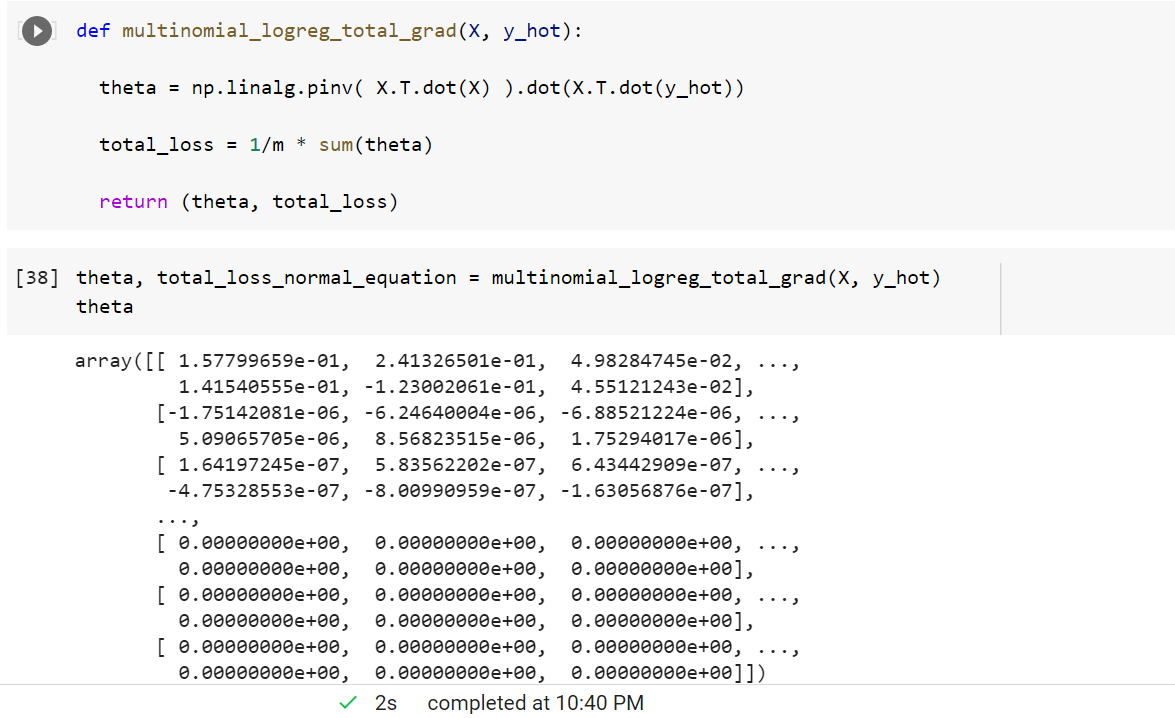
**Vậy thời gian chạy 10 vòng lặp với tham số chuẩn hóa L2 regularization cần 16s để chạy thuật toán với 10 vòng lặp** (Trên công cụ soạn thảo Google Colab đã tính toán thời gian thực thi thuật toán chúng ta trên hình trên). Với phương pháp trên, thời gian kỳ vọng để chạy **1000 vòng lặp** là: **26m 40s**.

Sau khi thực nghiệm, kết quả sau khi chạy **1000 vòng lặp** là:



## **Part 3.Tối ưu trong tính toán dữ liệu**

**Để tính toán hiệu quả hơn ta sử dụng ma trận số học Numpy thay vì vòng lặp for**

****

**Dựa trên kết quả thực nghiệm, có thể thấy khi dùng nhân ma trận số học Numpy thời gian chạy mất khoảng 2s để tính tham số tối ưu W.** Đây là phương pháp tìm nghiệm của bài toán Linear Regression mà không cần tới vòng lặp, không cần lựa chọn Learning Rate, và cũng không cần phải Scaling dữ liệu Trong khi ở part 2 như chúng tôi đã trình bày, **thời gian chạy 10 vòng lặp với tham số chuẩn hóa L2 regularization mất tới 16s để thực thi so với chỉ 2s là đã tính được nghiệm tối ưu ở phương pháp này.**

Sau đây chúng tôi sẽ đưa ra bảng số liệu so sánh giữa phương pháp Normal Equation và Gradient Descent như sau.

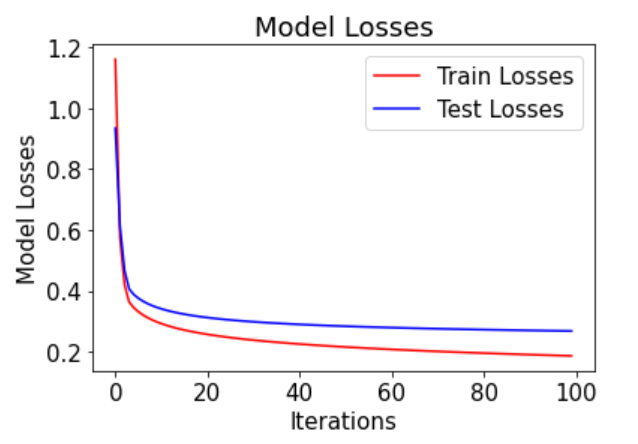
**Bảng So sánh giữa Normal Equation và Gradient Descent.**

|  |  |
| --- | --- |
| Gradient Descent | Normal Equation |
| Cần phải chọn Learning Rate | Không cần chọn Learning Rate |
| Cần nhiều vòng lặp | Không cần vòng lặp |
| Thời gian tính: *O(kn2)* | Thời gian tính: *O(n3)*, cần phải tính ma trận nghịch đảo |
| Hoạt động tốt với dữ liệu lớn | Rất chậm với dữ liệu lớn |

Với Normal Equation, việc tính toán khá mất thời gian *O(n3)* nên với dữ liệu lớn (n > 10.000 dữ liệu) chúng ta nên sử dụng Gradient Descent.

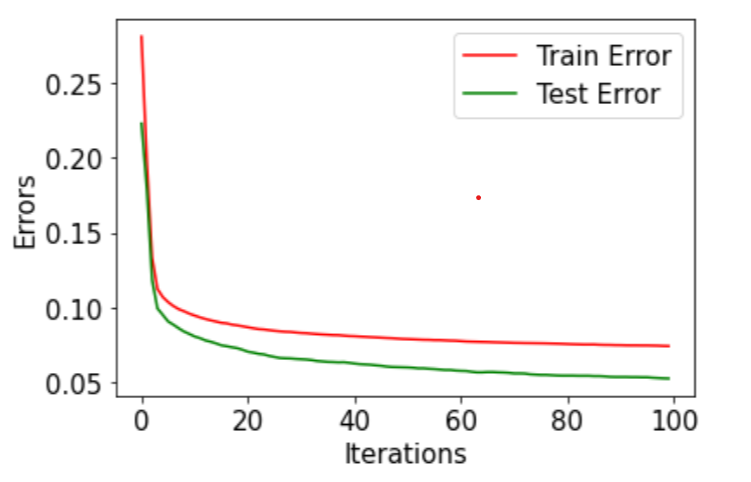
## **Part 4. Evaluating gradient descent**

Để đánh giá được chính xác liệu model của ta có thực sự dự đoán bao quát được cho cả mô hình hay không, liệu ta có rơi vào trường hợp overfitting. Chúng tôi sẽ sử dụng biểu đồ đường dây để đánh giá tập huấn luyện và tập test trong 1000 vòng lặp thực hiện thuật toán Gradient Descent như đã trình bày ở Part 2. Biểu đồ sẽ ghi lại kết quả tổng giá trị loss với mỗi 10 vòng lặp được thực thi.



Biểu đồ so sánh giá trị Loss giữa tập Train và Test

Như ta có thể thấy xu hướng trong biểu đồ trên là khi số lượng vòng lặp càng lớn cũng như số features đầu vào càng tăng đột ngột (khoảng 60.000 features) thì giá trị hàm Loss đối với Training Set và Test Set đều giảm đồng thời với nhau. Như vậy chúng tôi có thể kết luận khả năng cao đây là mô hình Good-fit với xác suất dự đoán khái quát được cả tập training và tập test.



Biểu đồ so sánh giá trị Error giữa tập Train và Test