## Các khái niệm, định lý

1. Gradient: là vector chứa các đạo hàm riêng của một hàm số đối với các biến đầu vào (đối với hàm số đơn biến, ta dùng khái niệm Derivative thay cho Gradient).

* Nó biểu thị hướng và mức độ tăng nhanh nhất của hàm số tại một điểm cụ thể.
* Trong Gradient descent: gradient được sử dụng để tìm hướng giảm nhanh nhất của hàm mục tiêu và ngược lại đối với Gradient ascent.

#### Descent | Ascent:

* Descent (descending): có nghĩa là sự đi xuống, (giảm dần).
* Ascent (ascending) : có nghĩa là sự đi lên (tăng dần).

Trong Gradient Descent, mục tiêu là tìm điểm cực tiểu của hàm mục tiêu, nên gradient được sử dụng để đi xuống dốc của hàm số. Ngược lại, trong Gradient Ascent, mục tiêu là tìm điểm cực đại của hàm mục tiêu, nên gradient được sử dụng để đi lên dốc của hàm số.

1. Learning rate: là một tham số quan trọng trong quá trình Gradient Descent/Ascent. Nó xác định độ lớn của bước di chuyển được thực hiện theo hướng gradient.

* Learning rate quá lớn có thể dẫn đến vượt qua điểm cực tiểu/ cực đại, khiến thuật toán không thể hội tụ được.
* Learning rate quá nhỏ thì có thể làm tăng thời gian hội tụ hay hội tụ ở một điểm không tối ưu.

1. Objective Function: là hàm số cần tối ưu hóa trong quá trình gradient descent/ ascent, là hàm mà chúng ta muốn tìm điểm cực tiểu/ cực đại.
2. Iterations: Quá trình gradient descent/ ascent thường được thực hiện thông qua nhiều vòng lặp. Mỗi vòng lặp bao gồm việc tính toán gradient của hàm mục tiêu tại một điểm và điều chỉnh các tham số dựa trên gradient đó cho đến khi đạt đến mục tiêu chí hội tụ nhất định (ví dụ đạt giá trị cực tiểu/ cực đại khi gradient gần như bằng 0).

## Giới thiệu:

Trong toán tối ưu, có nhiều phương pháp để tìm ra cực trị hàm số, cách phổ biến nhất là: Tìm đạo hàm, giải phương trình đạo hàm bằng 0, được hữu hạn các nghiệm , thay từng nghiệm vào hàm số cho ra giá trị cực trị tại đó rồi tìm điểm làm cho hàm đạt giá trị cực trị đó.

Tuy nhiên, trong nhiều trường hợp, việc giải phương trình đạo hàm bằng 0 là bất khả thi vì hàm số có đạo hàm phức tạp hay dữ liệu đa chiều… Vì thế người ta nghĩ ra cách để tìm cực trị dựa trên việc sử dụng gradient của hàm để di chuyển từng bước trong không gian biến, xuất phát từ một điểm mà chúng ta coi là gần với nghiệm của bài toán, sau đó dùng phép lặp để tiến gần hơn đến cực trị (đạo hàm gần đến 0) - đó là Gradient Descent và Gradient Ascent.

## Ứng dụng:

Phương pháp gradient descent và gradient ascent được áp dụng rộng rãi trong nhiều lĩnh vực khác nhau:

* Machine Learning/ Deep Learning: huấn luyện các mô hình, gradient descent/ascent được sử dụng để tối ưu hóa các hàm mất mát (loss function) và đạt được tham số tốt nhất cho mô hình bằng cách tính toán và điều chỉnh gradient.
* Tối ưu hóa không gian tham số: gradient descent/ascent được sử dụng để tối ưu hóa các hàm mục tiêu trong các bài toán tối ưu hóa không gian tham số. Ví dụ, trong tối ưu hóa tham số mô hình trong việc đánh giá hiệu suất tài chính, thiết kế mạch điện, tối ưu hóa tài nguyên trong mạng lưới điện, các phương pháp này có thể được áp dụng để tìm ra các bộ tham số tốt nhất để đạt được mục tiêu mong muốn.
* Trích xuất thông tin từ dữ liệu: Trong lĩnh vực Trí tuệ nhân tạo và xử lý dữ liệu, Gradient Descent và Gradient Ascent được sử dụng để tối ưu hóa các hàm mục tiêu trong việc trích xuất thông tin từ dữ liệu. Ví dụ, trong phân cụm (clustering), học không giám sát (unsupervised learning), và phân loại dữ liệu, các phương pháp này có thể được sử dụng để tìm ra các cụm tối ưu, các quy tắc phân loại tối ưu, hoặc các mô hình phân loại tối ưu.
* Xử lý ảnh và thị giác máy tính: Trong lĩnh vực xử lý ảnh và thị giác máy tính, Gradient Descent và Gradient Ascent được sử dụng để tối ưu hóa các hàm mục tiêu trong việc xử lý và phân tích ảnh. Ví dụ, trong việc điều chỉnh các thông số của một mô hình xử lý ảnh để tăng độ chính xác hoặc tối ưu hóa quá trình nhận dạng vật thể, các phương pháp này có thể được áp dụng để tìm ra các bộ tham số tốt nhất để đạt được kết quả tối ưu.

## Gradient Descent

## Thuật toán

Ý tưởng chung: điều chỉnh các tham số dựa trên đạo hàm (gradient) của hàm số đó.

Các bước chính của thuật toán:

1. Khởi tạo: bắt đầu việc khởi tạo giá trị ban đầu cho các tham số của mô hình.
2. Tính toán gradient: tính gradient của hàm mục tiêu đối với các tham số hiện tại.
3. Cập nhật tham số: sử dụng gradient tính được, điều chỉnh các tham số của mô hình bằng cách ***di chuyển ngược dấu với đạo hàm*** bằng một bước di chuyển được xác định bởi learning rate.
4. Lặp: lặp lại bước 2 và 3 cho đến khi đạt đến điều kiện dừng (thường là đạt được giá trị cực tiểu/ cực đại mong muốn, hoặc gradient gần như bằng 0).

Công thức:

θt+1 = θt − η∇θJ(θt)

Trong đó:

* + η : learning rate
  + θ : tập hợp các ttham số của mô hình
  + ∇θJ(θt) : đạo hàm của hàm mục tiêu tại θt
  + Dấu trừ thể hiện việc ta phải đi ngược với đạo hàm.

## Các thuật toán tối ưu Gradient Descent:

- Momentum

- Nesterov accelerated gradient (NAG)

- Các thuật toán khác: Adagrad, Adam, RMSprop, …

## Biến thể của Gradient Descent:

- Batch Gradient Descent

- Stochastic Gradient Descent

- Mini-batch Gradient Descent

## Điều kiện dừng:

Để biết khi nào thuật toán đã hội tụ và dừng lại, có vài phương pháp như sau:

1. Giới hạn số vòng lặp: đây là phương pháp phổ biến nhất và cũng để đảm bảo rằng chương trình chạy không quá lâu. Tuy nhiên, một nhược điểm của cách làm này là có thể thuật toán dừng lại trước khi đủ gần với nghiệm.
2. So sánh gradient của nghiệm tại hai lần cập nhật liên tiếp, khi nào giá trị này đủ nhỏ thì dừng lại. Phương pháp này cũng có một nhược điểm lớn là việc tính đạo hàm đôi khi trở nên quá phức tạp (ví dụ như khi có quá nhiều dữ liệu).
3. So sánh giá trị của hàm mất mát của nghiệm tại hai lần cập nhật liên tiếp, khi nào giá trị này đủ nhỏ thì dừng lại. Nhược điểm của phương pháp này là nếu tại một thời điểm, đồ thị hàm số có dạng bẳng phẳng tại một khu vực nhưng khu vực đó không chứa điểm local minimum (khu vực này thường được gọi là saddle points), thuật toán cũng dừng lại trước khi đạt giá trị mong muốn.
4. Trong SGD và mini-batch GD, cách thường dùng là so sánh nghiệm sau một vài lần cập nhật

## Ví dụ cụ thể ứng dụng gradient descent vào Recommender System :

* Mô tả dữ liệu đầu vào:
* Giải thích ngắn gọn các bước thực hiện ví dụ:
* Kết luận tính ứng dụng của thuật toán:

1. Recommender system (RS) là gì?

A picture containing text, screenshot, human face, online advertising

Description automatically generated

A screenshot of a social media account

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence

* Recommender system (hệ thống đề xuất) – là một nhánh của hệ thống lọc thông tin nhằm mục đích dự đoán “sở thích” của người dùng, từ đó đưa ra các đề xuất phù hợp cho người dùng dựa trên các thông tin đó. Mục tiêu của RS là cung cấp cho người dùng những nội dung, sản phẩm hoặc thông tin mà họ có khả năng quan tâm hoặc thích.
* Ví dụ:

+ Netflix: thông qua lịch sử xem phim/ đánh giá (rating) của người dùng, phân tích dựa trên các đặc trưng của film và của người dùng để đưa ra gợi ý các bộ phim tương tự, dự đoán những gì người dùng có thể muốn xem dựa trên hành vi của những người dùng khác…

+ Amazon: sử dụng lịch sử mua hàng của khách hàng để đưa ra đề xuất cho sản phẩm mới.

+ Các trang thương mại điện tử như Shopee, Lazada…: phân tích vật phẩm họ đã mua (sách, quần áo, thức ăn…) , đặc điểm của người dùng để đưa ra đề xuất sản phẩm có thể họ quan tâm.

+ Facebook: đề xuất kết bạn thông qua bạn chung, hiển thị những bài viết liên quan của các trang hay người dùng khác thông qua dữ liệu về tương tác cảm xúc hay bình luận của người dùng.

1. Các kiểu Recommender System ?

* Có nhiều kiểu Recommender System như:

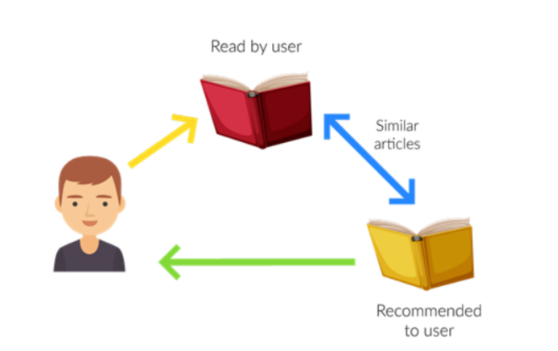
+ Collaborative Filtering (CF): tập trung vào các mối quan hệ giữa người dùng và mục tiêu -> dẫn đến vấn đề về độ thưa thớt, vì khó có thể tìm được đủ người đã đánh giá cho một mục tiêu nhất định.

A diagram of a person and person

Description automatically generated with low confidence

*Collaborative filtering recommender system*

+ Content-Based Filtering: tập trung vào các thuộc tính của mặt hàng để đưa ra đề xuất -> dẫn đến vấn đề về khả năng mở rộng, vì hệ thống cần được cập nhật liên tục với nội dung mới để đưa ra các đề xuất chính xác.

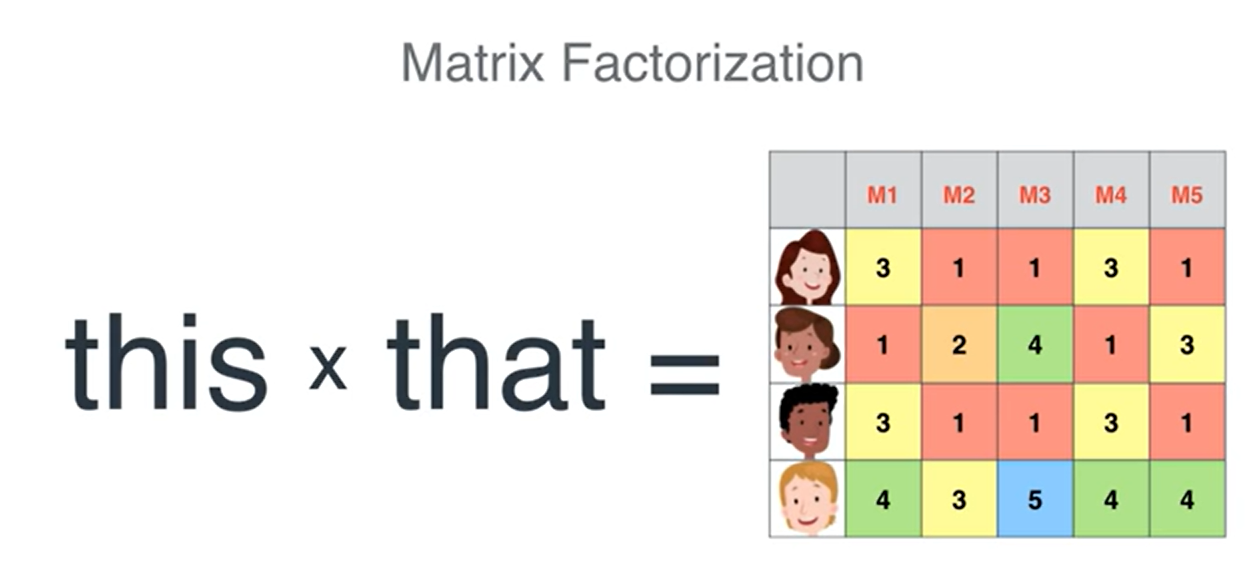


*Content-based recommender system*

+ Hybrid Recommender System: kết hợp hai phương pháp trên -> khắc phục được những hạn chế của hai phương pháp.

1. Thuật toán trong Recommender system kết hợp với gradient descent:

* Có nhiều thuật toán ML có thể được dùng trong RS, mỗi thuật toán có điểm mạnh và điểm yếu riêng, tùy vào đặc điểm dữ liệu và bài toán cụ thể.
* Sau đây ta sẽ xem xét áp dụng thuật toán Matrix Factorization vào bài toán hệ thống đề xuất phim cho người dùng, trong trường hợp này là dựa trên các đặc điểm (features) của phim và đánh giá, độ quan tâm của người dùng đối với các features đó



* Dùng phân tích ma trận, lúc này ta sẽ có ma trận đánh giá (ratings) của người dùng đối với mục tiêu (movies/films) sẽ được phân thành hai ma trận con:

+ Ma trận đặc trưng của người dùng (tạm gọi là M) thể hiện độ quan tâm của người dùng đối với các đặc trưng của film, với số hàng là số người dùng trong hệ thống và số cột là các đặc trưng (features)

+ Ma trận đặc trưng của film (tạm gọi là N) biểu diễn mức độ/ điểm mà film cung cấp cho từng đặc trưng, với số hàng là số film, các cột thể hiện các đặc trưng (features)

+ Các đặc trưng như về đạo diễn, diễn viên, đánh giá trung bình, thể loại,… Trong ví dụ này ta sẽ xét hai đặc trưng cụ thể là về thể loại film: hành động (action) và hài hước (comedy).

A picture containing text, screenshot

Description automatically generated

* Khi đó các đánh giá dự đoán sẽ là tích MNT.
* Vấn đề đặt ra là về bộ nhớ lưu trữ (storage) cho ma trận tích này, giả sử có 1000 movies và 2000 user thì sẽ phải dành ra 2.000.000 params để lưu trữ. Và trên thực tế thì máy tính không thể biết được người nào thích comedy, người nào thích action và nó phải thống kê để tìm ra các features, những features mà người dùng thích và điểm số (scores) mà movies cung cấp cho từng feature đó. Do đó, ta áp dụng machine learning để huấn luyện mô hình để tìm ra các tham số phù hợp.

A picture containing screenshot, text, human face

Description automatically generated

* Ta sẽ bắt đầu với các giá trị ngẫu nhiên cho ma trận M và N, sau đó tính tích hai MNT và so sánh với ma trận gốc (ratings):

A screenshot of a grid with a grid with a grid with a grid with a grid with a grid with a grid with a grid with a grid with a grid with a grid with a grid

Description automatically generated with low confidence

* Khi này, mục tiêu của ta là sẽ tìm các ma trận đặc trưng này sao cho tích của chúng xấp xỉ lại ma trận đánh giá ban đầu một cách tốt nhất, bằng cách điều chỉnh các tham số của hai ma trận.

A grid with a grid with a grid with a grid with a grid with a grid with a grid with a grid with a grid with a grid with a grid with a grid with a grid with

Description automatically generated with low confidence

* Khi này ta sẽ tính giá trị hàm mất mát để đo lường sự sai khác giữa giá trị dự đoán và giá trị thực tế, và tìm cách giảm thiểu giá trị này.
* Một phương pháp phổ biến được áp dụng là gradient descent, ta sẽ dùng gradient descent để điều chỉnh các giá trị trong ma trận đặc trưng của người dùng và mục tiêu để giảm thiểu sai số giữa ma trận xấp xỉ (tích hai ma trận đặc trưng) và ma trận đánh giá thực tế từ người dùng.

### Ý tưởng thực hiện như sau:

1. Tạo dữ liệu : tạo bảng đánh giá từ người dùng cho các bộ phim với các đánh giá ngẫu nhiên từ 0 -> 5, sau đó ghi vào file `ratings.txt`, tương tự tạo hai ma trận đặc trưng con `user\_features` và `movie\_features` và ghi vào file text cùng tên, ở ví dụ này, ta sẽ tạo bảng dữ liệu kích thước 501 \* 100 (users x movies), trong đó người dùng cuối (thứ 501) được tạo cố định chỉ mới đánh giá 3 bộ phim, hai ma trận đặc trưng con có kích thước là 501 x 10 và 100 x 10 , trong trường hợp này giả sử có 10 đặc trưng (features).
2. Đọc dữ liệu từ file để chuẩn bị huấn luyện mô hình.
3. Chuẩn hóa dữ liệu bảng đánh giá (ratings): lần lượt trừ các giá trị dữ liệu cho giá trị trung bình (mean) của dữ liệu, mục đích của việc này là để:

* Tránh ảnh hưởng của quy mô dữ liệu: Nếu các đặc trưng trong dữ liệu có quy mô khác nhau (ví dụ: một đặc trưng có giá trị trong khoảng 0-1000, trong khi đặc trưng khác có giá trị trong khoảng 0-1), việc không chuẩn hóa dữ liệu có thể làm cho mô hình tập trung vào các đặc trưng có quy mô lớn hơn và bỏ qua những đặc trưng có quy mô nhỏ hơn. Điều này có thể gây ra sự thiếu cân bằng trong quá trình huấn luyện và dẫn đến mô hình không hoạt động tốt.
* Tăng tốc độ huấn luyện: Chuẩn hóa dữ liệu có thể giúp tối ưu hóa quá trình huấn luyện. Thuật toán máy học (như gradient descent) hoạt động tốt hơn trên dữ liệu có phân bố gần với phân phối chuẩn. Bằng cách chuẩn hóa dữ liệu, ta có thể đạt được phân phối gần với phân phối chuẩn và giúp thuật toán huấn luyện nhanh hơn.

1. Viết hàm tính giá tri mất mát trung bình cho mô hình dự đoán bằng cách tính tổng bình phương của hiệu giữa giá trị dự đoán và giá trị thực tế.
2. Viết hàm thực hiện thuật toán gradient descent để tối ưu hóa hai ma trận đặc trưng con, cơ bản qua mỗi vòng lặp sẽ thực hiện các bước sau:

* Tính độ lệch (difference) giữa giá trị dự đoán và giá trị thực tế
* Tính gradient: sử dụng công thức tính gradient từ đạo hàm riêng của hàm mất mát, công thức này sử dụng độ lệch và hai ma trận đặc trưng.
* Cập nhật `user\_features` và `movie\_features` bằng cách trừ cho (learning\_rate \* gradient tương ứng)
* Kiểm tra điều kiện dừng: nếu thỏa điều kiện thì thoát vòng lặp

1. Thiết lập các tham số learning rate và số vòng lặp và tiến hành huấn luyện mô hình.
2. Tính ứng dụng của gradient descent vào RS:

* Huấn luyện mô hình: GD dùng để tối ưu các tham số của mô hình như ma trận đặc trưng của người dùng và ma trận đặc trưng mục tiêu.
* Tối ưu hàm mất mát: trong RS ta cần đo lường sự khác biệt giữa giá trị dự đoán và giá trị thực tế của đánh giá. GD được dùng để tối ưu hóa hàm mất mát, giảm thiểu sai số giữa các giá trị dự đoán và giá trị thực tế.
* Điều chỉnh overfitting: GD có thể được dùng để điều chỉnh tham số như hệ số điều chỉnh (regularization) để giảm hiện tượng overfitting trong mô hình. Overfitting xảy ra khi mô hình quá phức tạp và hiệu suất trên dữ liệu huấn luyện cao nhưng không tổng quát tốt cho dữ liệu mới.
* Quay lại ví dụ trên, giả sử dưới đây là bảng đánh giá của người dùng cho các bộ phim họ đã xem, các khoảng trống là các phim chưa xem:

A picture containing human face, screenshot, person, square

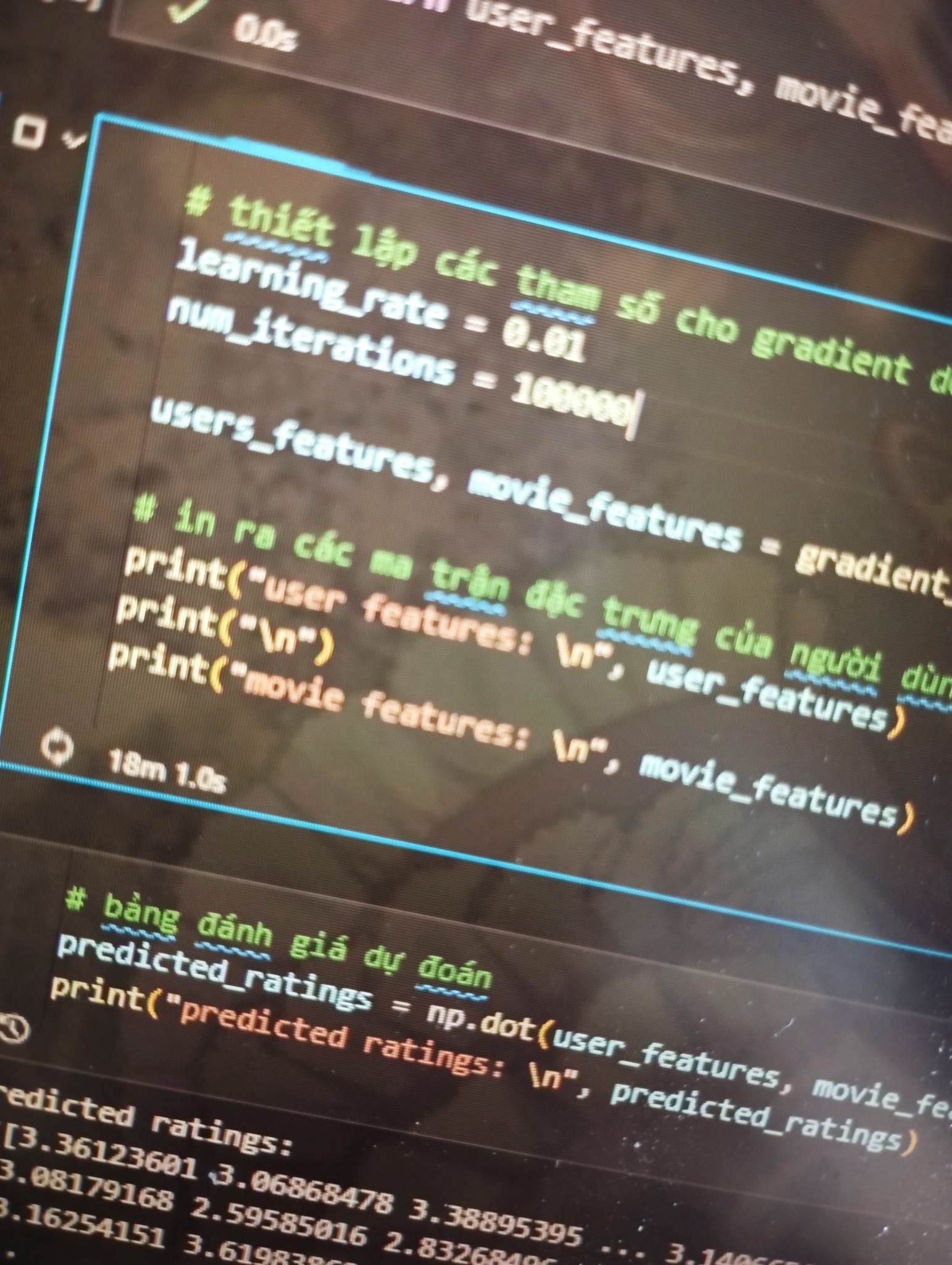
Description automatically generated

* Khi này ta có thể biết được giá trị ở các ô trống bằng cách nhân tích các vector hàng và cột tương ứng. Giả sử có 1 người D, họ đã xem qua các phim M2, M4, M5 và chưa xem M1, M3. Khi D quay lại hệ thống sẽ dự đoán được và đề xuất cho D xem phim M3 vì dự đoán của M3 (5) lớn hơn dự đoán của M1(4).
* A screenshot of a game

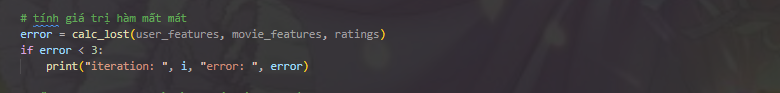
  Description automatically generated with medium confidence

## \*\*\* Quá trình chạy code:

* Thời gian đầu thì cứ thay đổi vô tội vạ không căn cứ các tham số learning\_rate thành 0.1, 0.01, 0.001 … rồi num\_iterations thành 100, 1000, 10000, 100000 hay thậm chí 1.000.000
* Thời gian chạy lâu với số lần lặp lớn thậm chí tới tận hơn 18 phút :”)



* Tuy vậy nhưng kết quả vẫn không như mong đợi, hai ma trận đặc trưng vẫn không được tối ưu, nên là tiến hành chạy và kiểm tra từ từ - thay đổi tham số learning rate với số lặp nhỏ tầm 100 – 1000, sau đó, qua mỗi vòng lặp sẽ in ra giá trị sai số, nhìn nhận sự thay đổi qua từng vòng lặp với từng giá trị tham số đầu vào và thay đổi dần dần giá trị learning rate để giá trị mất mát nhỏ hơn (tầm 1. – 2.), rồi sau đó cố định learning rate đó và tăng số vòng lặp lên.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

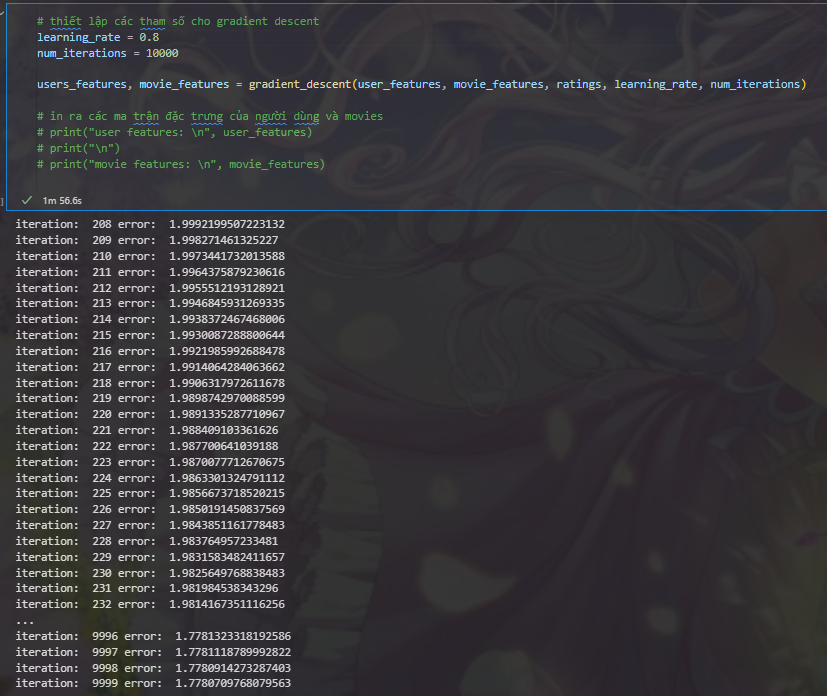
Description automatically generated

A screenshot of a computer

Description automatically generated

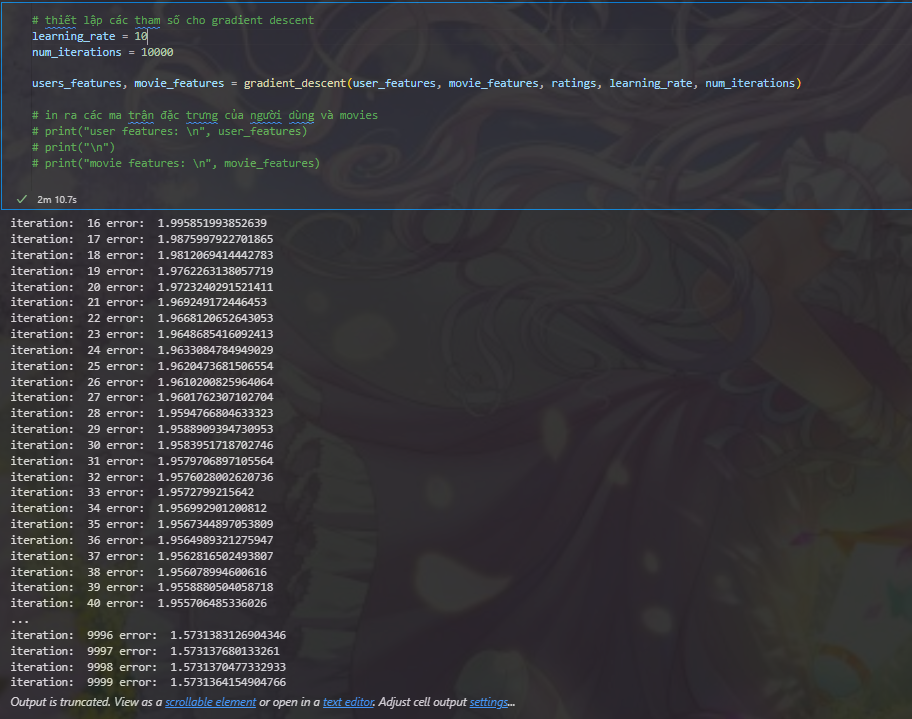
A screenshot of a computer

Description automatically generated



A screenshot of a computer

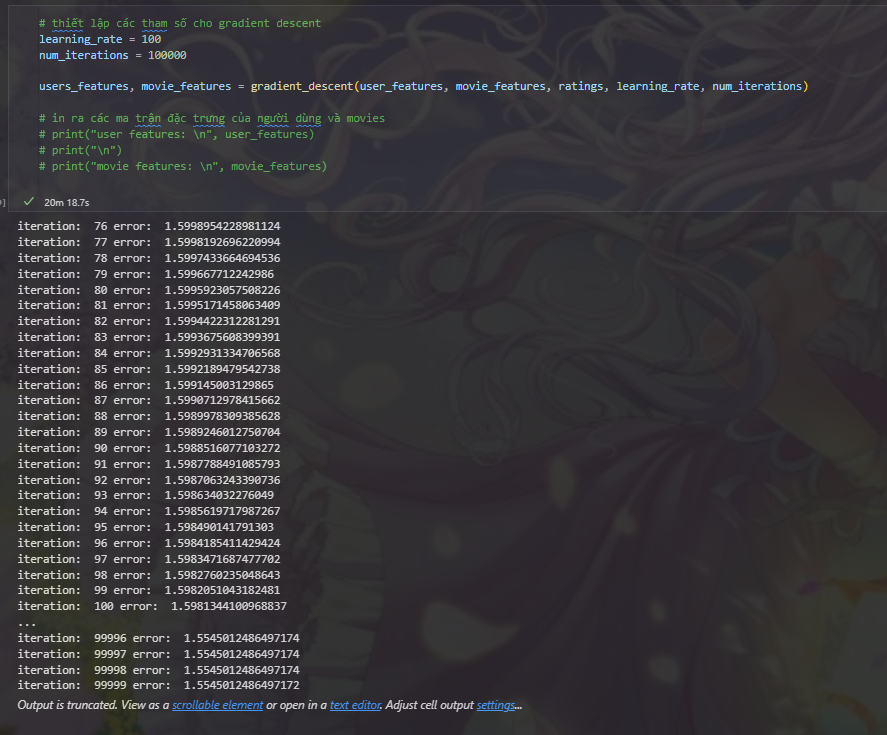
Description automatically generated



* Thay đổi code 1 tí và chờ :”) cũng rảnh mà :”)

A screenshot of a computer screen

Description automatically generated with medium confidence



* Thôi mệt quá kệ nó luôn nha :”))

## Tài liệu tham khảo

* [Machine Learning cơ bản (machinelearningcoban.com)](https://machinelearningcoban.com/2017/01/12/gradientdescent/)
* [(53) Gradient Descent, Step-by-Step - YouTube](https://www.youtube.com/watch?v=sDv4f4s2SB8)
* [4. Gradient Descent | Quy's blog (ndquy.github.io)](https://ndquy.github.io/posts/gradient-descent-2/)
* [Machine Learning cơ bản (machinelearningcoban.com)](https://machinelearningcoban.com/2017/01/16/gradientdescent2/#-ket-luan)
* [learn-co-students/gradient-descent-data-science-intro-000 (github.com)](https://github.com/learn-co-students/gradient-descent-data-science-intro-000)
* [mattnedrich/GradientDescentExample: Example demonstrating how gradient descent may be used to solve a linear regression problem (github.com)](https://github.com/mattnedrich/GradientDescentExample)
* [Machine-Learning-with-Python/Logistic Regression in Python - Step by Step.ipynb at master · susanli2016/Machine-Learning-with-Python · GitHub](https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/Logistic%20Regression%20in%20Python%20-%20Step%20by%20Step.ipynb)
* [Gradient Ascent: When to use it in machine learning? (analyticsindiamag.com)](https://analyticsindiamag.com/gradient-ascent-when-to-use-it-in-machine-learning/)
* [What Is the Difference Between Gradient Descent and Gradient Ascent? | Baeldung on Computer Science](https://www.baeldung.com/cs/gradient-descent-vs-ascent)
* [machine learning - Gradient Ascent vs Gradient Descent in Logistic Regression - Cross Validated (stackexchange.com)](https://stats.stackexchange.com/questions/258721/gradient-ascent-vs-gradient-descent-in-logistic-regression)
* [Recommender Systems in Machine Learning: Examples - Data Analytics (vitalflux.com)](https://vitalflux.com/recommender-systems-in-machine-learning-examples/)
* [Featured Hybrid Recommendation System Using Stochastic Gradient Descent | Atlantis Press (atlantis-press.com)](https://www.atlantis-press.com/journals/ijndc/125950416/view)
* [(71) How does Netflix recommend movies? Matrix Factorization - YouTube](https://www.youtube.com/watch?v=ZspR5PZemcs)