

## 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.

### 2. 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.

3. **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.
4. **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.
5. **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

### 1. 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:

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t)$$

Trong đó:

- $\eta$  : learning rate
- $\theta$  : tập hợp các tham số của mô hình
- $\nabla J(\theta_t)$  : đạo hàm của hàm mục tiêu tại  $\theta_t$
- Dấu trừ thể hiện việc ta phải đi ngược với đạo hàm.

## 2. Các thuật toán tối ưu Gradient Descent:

- Momentum
- Nesterov accelerated gradient (NAG)
- Các thuật toán khác: Adagrad, Adam, RMSprop, ...

## 3. Biến thể của Gradient Descent:

- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini-batch Gradient Descent

## 4. Điều kiện dừng:

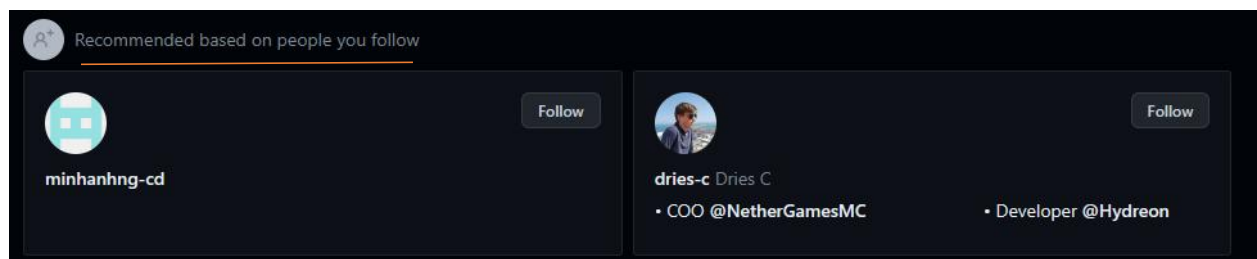
Để biết khi nào thuật toán đã hội tụ và dừng lại, có vài phương pháp như sau:

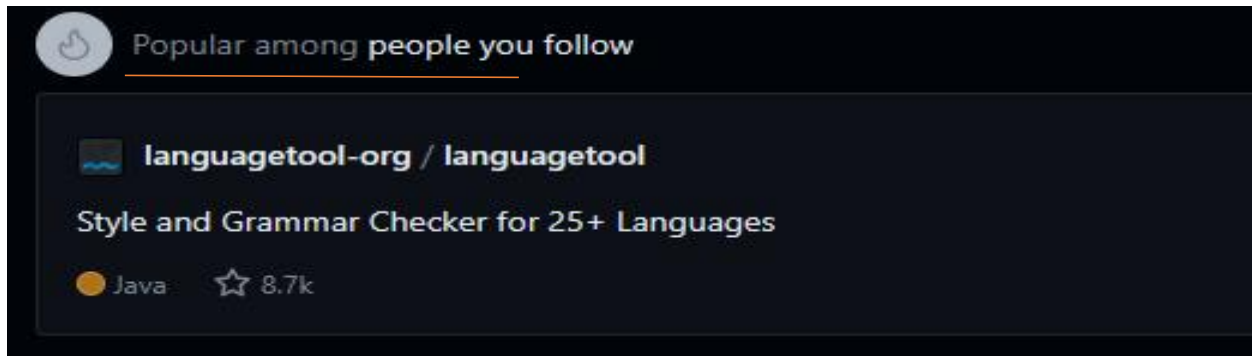
1. Giới hạn số vòng lặp: đây là phương pháp phổ biến nhất và cũng để đảm bảo rằng chương trình chạy không quá lâu. Tuy nhiên, một nhược điểm của cách làm này là có thể thuật toán dừng lại trước khi đủ gần với nghiệm.
2. So sánh gradient của nghiệm tại hai lần cập nhật liên tiếp, khi nào giá trị này đủ 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 bằng 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

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

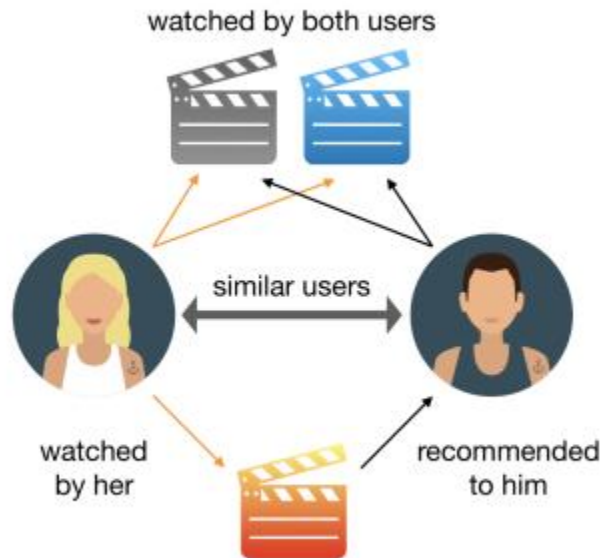
- Mô tả dữ liệu đầu vào:
- Giải thích ngắn gọn các bước thực hiện ví dụ:
- Kết luận tính ứng dụng của thuật toán:

### 1. Recommender system (RS) là gì?



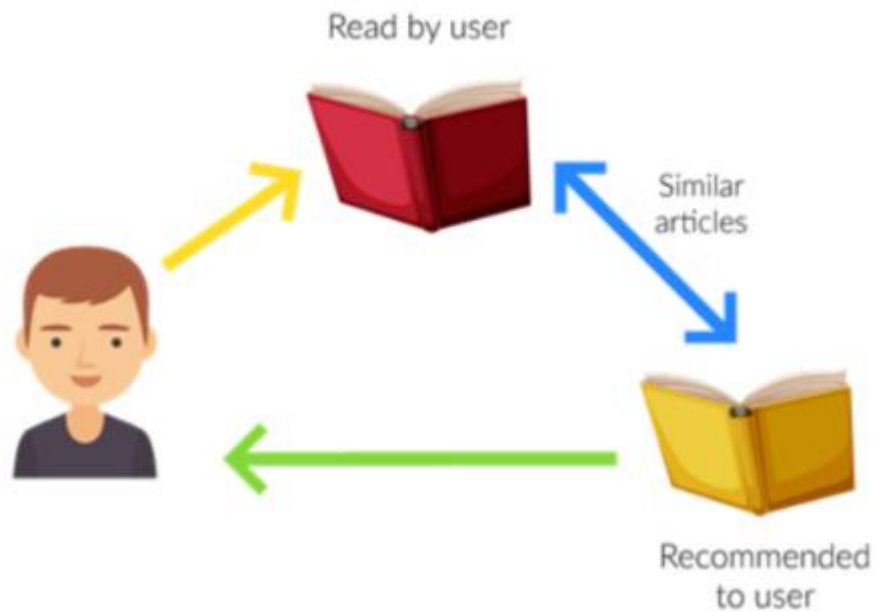


- Recommender system (hệ thống đề xuất) – là một nhánh của hệ thống lọc thông tin nhằm mục đích dự đoán “sở thích” của người dùng, từ đó đưa ra các đề xuất phù hợp cho người dùng dựa trên các thông tin đó. Mục tiêu của RS là cung cấp cho người dùng những nội dung, sản phẩm hoặc thông tin mà họ có khả 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.
2. 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.



*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.

3. 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

-

## Matrix Factorization

this x that =

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4



- 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).

# Matrix Factorization

	M1	M2	M3	M4	M5
 Comedy	3	1	1	3	1
 Action	1	2	4	1	3



	 Comedy	 Action
 A		
 B		
 C		
 D		

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4







- Khi đó các đánh giá dự đoán sẽ là tích  $MN^T$ .
- 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.







# Matrix Factorization

	M1	M2	M3	M4	M5
 Comedy	3	1	1	3	1
 Action	1	2	4	1	3







	 Comedy	 Action
 A	1	0
 B	0	1
 C	1	0
 D	1	1

	M1	M2	M3	M4	M5
 A	3	1	1	3	1
 B	1	2	4	1	3
 C	3	1	1	3	1
 D	4	3	5	4	4





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

	M1	M2	M3	M4	M5
F1	1.2	3.1	0.3	2.5	0.2
F2	2.4	1.5	4.4	0.4	1.1





  

	F1	F2
 A	0.2	0.5
 B	0.3	0.4
 C	0.7	0.8
 D	0.4	0.5

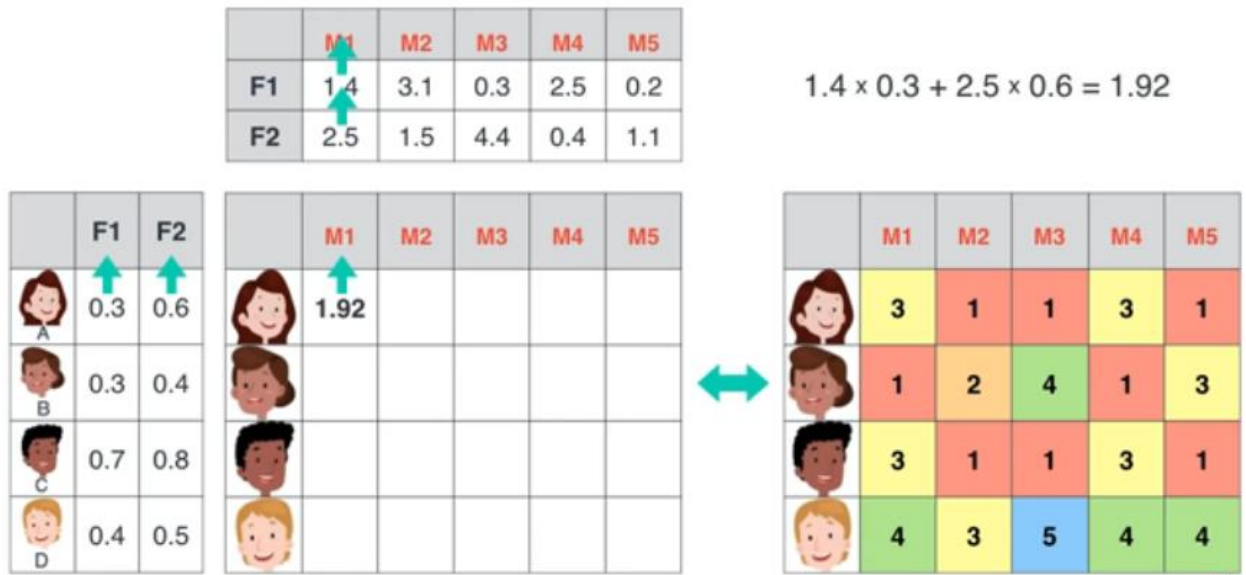
  

	M1	M2	M3	M4	M5
 A	1.44	1.37	2.26	0.7	0.59
 B	1.32	1.53	1.85	0.91	0.5
 C	2.76	3.37	3.73	2.07	1.02
 D	1.68	1.99	2.32	1.2	0.63

	M1	M2	M3	M4	M5
 A	3	1	1	3	1
 B	1	2	4	1	3
 C	3	1	1	3	1
 D	4	3	5	4	4

- 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.



- 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.
4. Viết hàm tính giá trị 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.
  5. 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 ( $\text{learning\_rate} * \text{gradient}$  tương ứng)
    - Kiểm tra điều kiện dừng: nếu thỏa điều kiện thì thoát vòng lặp
  6. Thiết lập các tham số learning rate và số vòng lặp và tiến hành huấn luyện mô hình.

4. 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:

	M1	M2	M3	M4	M5
A	3		1		1
B	1		4	1	
C	3	1		3	1
D		3		4	4

- 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).

	M1	M2	M3	M4	M5
F1	3	1	1	3	1
F2	1	2	4	1	3

	F1	F2
A	1	0
B	0	1
C	1	0
D	1	1

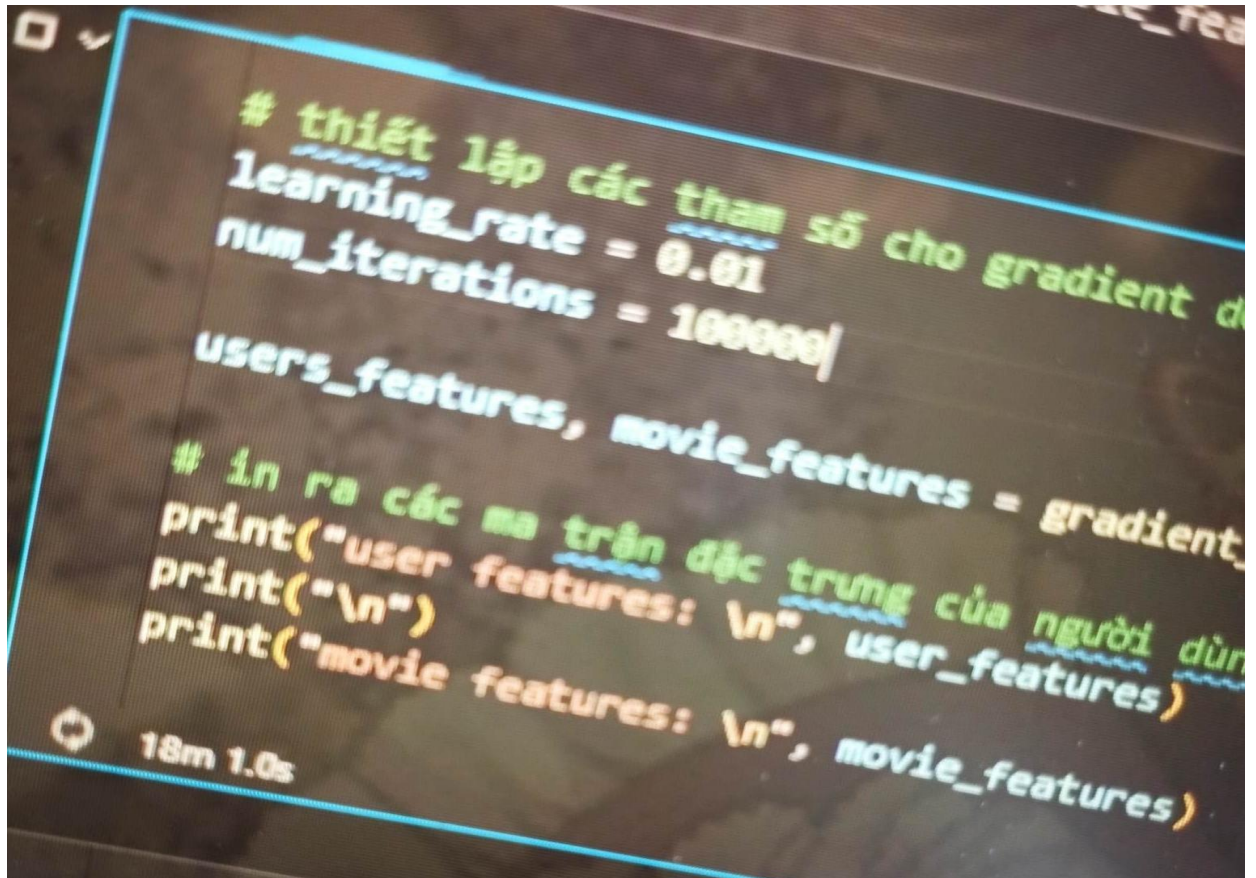
	M1	M2	M3	M4	M5
A	3	1	1	3	1
B	1	2	4	1	3
C	3	1	1	3	1
D	4	3	5	4	4



M3

\*\*\* 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 :")



```
# thiết lập các tham số cho gradient d
learning_rate = 0.01
num_iterations = 100000|

users_features, movie_features = gradient

# in ra các ma trận đặc trưng của người dùng
print("user features: \n", user_features)
print("\n")
print("movie features: \n", movie_features)
```

18m 1.0s

- 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.



```

# tính giá trị hàm mất mát
error = calc_lost(user_features, movie_features, ratings)
if error < 3:
    print("iteration: ", i, "error: ", error)

# thiết lập các tham số cho gradient descent
learning_rate = 0.5
num_iterations = 1000

users_features, movie_features = gradient_descent(user_features, movie_features, ratings, learning_rate, num_iterations)

# in ra các ma trận đặc trưng của người dùng và movies
# print("user features: \n", user_features)
# print("\n")
# print("movie features: \n", movie_features)

```

✓ 15.2s

```

iteration: 117 error: 2.9857582229946362
iteration: 118 error: 2.9689703074753364
iteration: 119 error: 2.9524772877859093
iteration: 120 error: 2.9362733276271245
iteration: 121 error: 2.9203527242945313
iteration: 122 error: 2.9047099050852623
iteration: 123 error: 2.8893394238181056
iteration: 124 error: 2.874235957462716
iteration: 125 error: 2.8593943028740343
iteration: 126 error: 2.844809373628127
iteration: 127 error: 2.830476196955807
iteration: 128 error: 2.816389910770573
iteration: 129 error: 2.802545760787494
iteration: 130 error: 2.7889390977298643
iteration: 131 error: 2.775565374620515
iteration: 132 error: 2.7624201441548455
iteration: 133 error: 2.7494990561527257
iteration: 134 error: 2.7367978550865297
iteration: 135 error: 2.7243123776826934
iteration: 136 error: 2.7120385505942606
iteration: 137 error: 2.6999723881420032
iteration: 138 error: 2.6881099901217786
iteration: 139 error: 2.6764475396758893
iteration: 140 error: 2.6649813012262853
iteration: 141 error: 2.6537076184675414
...
iteration: 996 error: 1.932998468108441
iteration: 997 error: 1.9329916498823314
iteration: 998 error: 1.9329848326266708
iteration: 999 error: 1.9329780163206773

```

```
# thiết lập các tham số cho gradient descent
learning_rate = 1
num_iterations = 1000

users_features, movie_features = gradient_descent(user_features, movie_features, ratings, learning_rate, num_iterations)

# in ra các ma trận đặc trưng của người dùng và movies
# print("user features: \n", user_features)
# print("\n")
# print("movie features: \n", movie_features)
```

✓ 13.2s

```
iteration: 55 error: 2.9729151736322526
iteration: 56 error: 2.9389024371413712
iteration: 57 error: 2.906126539302136
iteration: 58 error: 2.8745370305579354
iteration: 59 error: 2.844085820918475
iteration: 60 error: 2.8147270516072656
iteration: 61 error: 2.786416974829991
iteration: 62 error: 2.759113841073817
iteration: 63 error: 2.73277793396019
iteration: 64 error: 2.7073707682041825
iteration: 65 error: 2.682856402070257
iteration: 66 error: 2.6591999441571246
iteration: 67 error: 2.636368173869551
iteration: 68 error: 2.614329323371695
iteration: 69 error: 2.593053004641025
iteration: 70 error: 2.5725101407537867
iteration: 71 error: 2.5526729011203004
iteration: 72 error: 2.5335146404096123
iteration: 73 error: 2.5150098409223665
iteration: 74 error: 2.4971340581886534
iteration: 75 error: 2.4798638695838986
iteration: 76 error: 2.4631768257708875
iteration: 77 error: 2.447051404789826
iteration: 78 error: 2.4314669686310424
iteration: 79 error: 2.4164037221366117
...
iteration: 996 error: 1.936863312186352
iteration: 997 error: 1.936848988880606
iteration: 998 error: 1.9368346624254633
iteration: 999 error: 1.9368203328205895
```

```
# thiết lập các tham số cho gradient descent
learning_rate = 0.8
num_iterations = 1000

users_features, movie_features = gradient_descent(user_features, movie_features, ratings, learning_rate, num_iterations)

# in ra các ma trận đặc trưng của người dùng và movies
# print("user features: \n", user_features)
# print("\n")
# print("movie features: \n", movie_features)
```

✓ 14.6s

```
iteration: 202 error: 1.9998540656450077
iteration: 203 error: 1.9987150629242305
iteration: 204 error: 1.99760118728136
iteration: 205 error: 1.9965118583366506
iteration: 206 error: 1.9954465097402232
iteration: 207 error: 1.9944045888135904
iteration: 208 error: 1.9933855562009688
iteration: 209 error: 1.9923888855300997
iteration: 210 error: 1.9914140630822799
iteration: 211 error: 1.9904605874713535
iteration: 212 error: 1.9895279693313834
iteration: 213 error: 1.9886157310127601
iteration: 214 error: 1.9877234062865028
iteration: 215 error: 1.9868505400565004
iteration: 216 error: 1.985996688079488
iteration: 217 error: 1.9851614166925078
iteration: 218 error: 1.9843443025476584
iteration: 219 error: 1.983544932353913
iteration: 220 error: 1.9827629026258078
iteration: 221 error: 1.9819978194388066
iteration: 222 error: 1.9812492981911392
iteration: 223 error: 1.9805169633719504
iteration: 224 error: 1.9798004483355556
iteration: 225 error: 1.9790993950816527
iteration: 226 error: 1.978413454041307
...
iteration: 996 error: 1.9393198047560598
iteration: 997 error: 1.9393086993328315
iteration: 998 error: 1.9392975918521607
iteration: 999 error: 1.939286482313759
```



```
# thiết lập các tham số cho gradient descent
learning_rate = 0.3
num_iterations = 10000

users_features, movie_features = gradient_descent(user_features, movie_features, ratings, learning_rate, num_iterations)

# in ra các mã trần đặc trưng của người dùng và movies
# print("user features: \n", user_features)
# print("\n")
# print("movie features: \n", movie_features)
```

✓ 1m 52.0s

```
iteration: 540 error: 1.999745742069821
iteration: 541 error: 1.9993496022102397
iteration: 542 error: 1.998956761162141
iteration: 543 error: 1.9985671896958919
iteration: 544 error: 1.998180858855845
iteration: 545 error: 1.997797739957606
iteration: 546 error: 1.99741780458534
iteration: 547 error: 1.9970410245890993
iteration: 548 error: 1.9966673720821828
iteration: 549 error: 1.9962968194385198
iteration: 550 error: 1.9959293392900885
iteration: 551 error: 1.9955649045243533
iteration: 552 error: 1.9952034882817369
iteration: 553 error: 1.9948450639531106
iteration: 554 error: 1.9944896051773224
iteration: 555 error: 1.9941370858387382
iteration: 556 error: 1.9937874800648216
iteration: 557 error: 1.9934407622237287
iteration: 558 error: 1.9930969069219338
iteration: 559 error: 1.9927558890018804
iteration: 560 error: 1.992417683539652
iteration: 561 error: 1.9920822658426756
iteration: 562 error: 1.9917496114474402
iteration: 563 error: 1.9914196961172437
iteration: 564 error: 1.9910924958399658
...
iteration: 9996 error: 1.903422266648697
iteration: 9997 error: 1.903415927523606
iteration: 9998 error: 1.9034095881377242
iteration: 9999 error: 1.9034032484910661
```

```

# thiết lập các tham số cho gradient descent
learning_rate = 0.8
num_iterations = 10000

users_features, movie_features = gradient_descent(user_features, movie_features, ratings, learning_rate, num_iterations)

# in ra các ma trận đặc trưng của người dùng và movies
# print("user features: \n", user_features)
# print("\n")
# print("movie features: \n", movie_features)

```

✓ 1m 56.6s

```

iteration: 208 error: 1.9992199507223132
iteration: 209 error: 1.998271461325227
iteration: 210 error: 1.9973441732013588
iteration: 211 error: 1.9964375879230616
iteration: 212 error: 1.9955512193128921
iteration: 213 error: 1.9946845931269335
iteration: 214 error: 1.9938372467468006
iteration: 215 error: 1.9930087288800644
iteration: 216 error: 1.9921985992688478
iteration: 217 error: 1.9914064284063662
iteration: 218 error: 1.9906317972611678
iteration: 219 error: 1.9898742970088599
iteration: 220 error: 1.9891335287710967
iteration: 221 error: 1.988409103361626
iteration: 222 error: 1.987700641039188
iteration: 223 error: 1.9870077712670675
iteration: 224 error: 1.9863301324791112
iteration: 225 error: 1.9856673718520215
iteration: 226 error: 1.9850191450837569
iteration: 227 error: 1.9843851161778483
iteration: 228 error: 1.983764957233481
iteration: 229 error: 1.9831583482411657
iteration: 230 error: 1.9825649768838483
iteration: 231 error: 1.981984538343296
iteration: 232 error: 1.9814167351116256
...
iteration: 9996 error: 1.7781323318192586
iteration: 9997 error: 1.7781118789992822
iteration: 9998 error: 1.7780914273287403
iteration: 9999 error: 1.7780709768079563

```

```
# thiết lập các tham số cho gradient descent
learning_rate = 2
num_iterations = 10000

users_features, movie_features = gradient_descent(user_features, movie_features, ratings, learning_rate, num_iterations)

# in ra các ma trận đặc trưng của người dùng và movies
# print("user features: \n", user_features)
# print("\n")
# print("movie features: \n", movie_features)
```

✓ 1m 43.8s

```
iteration: 81 error: 1.9983648304021087
iteration: 82 error: 1.99606812307474
iteration: 83 error: 1.9938960634977425
iteration: 84 error: 1.9918414786466507
iteration: 85 error: 1.989897629875233
iteration: 86 error: 1.9880581851327572
iteration: 87 error: 1.986317193065278
iteration: 88 error: 1.9846690588648512
iteration: 89 error: 1.9831085217410906
iteration: 90 error: 1.981630633899173
iteration: 91 error: 1.980230740917236
iteration: 92 error: 1.9789044634242416
iteration: 93 error: 1.977647679986855
iteration: 94 error: 1.9764565111207346
iteration: 95 error: 1.9753273043479513
iteration: 96 error: 1.9742566202280336
iteration: 97 error: 1.9732412192955147
iteration: 98 error: 1.9722780498417363
iteration: 99 error: 1.971364236483238
iteration: 100 error: 1.9704970694632182
iteration: 101 error: 1.9696739946364181
iteration: 102 error: 1.9688926040913601
iteration: 103 error: 1.9681506273671305
iteration: 104 error: 1.9674459232249675
iteration: 105 error: 1.9667764719377063
...
iteration: 9996 error: 1.6244117771528988
iteration: 9997 error: 1.6244027231539069
iteration: 9998 error: 1.6243936716252143
iteration: 9999 error: 1.6243846225659238
```

```
# thiết lập các tham số cho gradient descent
learning_rate = 10
num_iterations = 10000

users_features, movie_features = gradient_descent(user_features, movie_features, ratings, learning_rate, num_iterations)

# in ra các ma trận đặc trưng của người dùng và movies
# print("user features: \n", user_features)
# print("\n")
# print("movie features: \n", movie_features)
```

✓ 2m 10.7s

```
iteration: 16 error: 1.995851993852639
iteration: 17 error: 1.9875997922701865
iteration: 18 error: 1.9812069414442783
iteration: 19 error: 1.9762263138057719
iteration: 20 error: 1.9723240291521411
iteration: 21 error: 1.969249172446453
iteration: 22 error: 1.9668120652643053
iteration: 23 error: 1.9648685416092413
iteration: 24 error: 1.9633084784949029
iteration: 25 error: 1.9620473681506554
iteration: 26 error: 1.9610200825964064
iteration: 27 error: 1.9601762307102704
iteration: 28 error: 1.9594766804633323
iteration: 29 error: 1.9588909394730953
iteration: 30 error: 1.9583951718702746
iteration: 31 error: 1.9579706897105564
iteration: 32 error: 1.9576028002620736
iteration: 33 error: 1.9572799215642
iteration: 34 error: 1.956992901200812
iteration: 35 error: 1.9567344897053809
iteration: 36 error: 1.9564989321275947
iteration: 37 error: 1.9562816502493807
iteration: 38 error: 1.956078994600616
iteration: 39 error: 1.9558880504058718
iteration: 40 error: 1.955706485336026
...
iteration: 9996 error: 1.5731383126904346
iteration: 9997 error: 1.573137680133261
iteration: 9998 error: 1.5731370477332933
iteration: 9999 error: 1.5731364154904766
```

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...

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



```

# hàm gradient descent
def gradient_descent(user_features, movie_features, ratings, learning_rate, num_iterations):
    # ma trận did_rate có giá trị 1 tại các vị trí có ratings > 0, ngược lại là 0
    did_rate = (ratings != 0) * 1
    # print("did_rate: \n", did_rate)

    # số đánh giá
    numRated = 0
    for i in range(did_rate.shape[0]):
        for j in range(did_rate.shape[1]):
            if did_rate[i, j] == 1:
                numRated += 1

    # lặp lại quá trình gradient descent
    for i in range(num_iterations):
        # tính độ lệch giữa dữ liệu thực tế và dữ liệu dự đoán
        predicted = np.dot(user_features, movie_features.T)
        # print("predicted: \n", predicted)
        difference = (ratings - predicted) * did_rate
        # print("difference: \n", difference)

        # tính gradient cho user_features và movie_features
        user_features_grad = -2/(numRated) * np.dot(difference, movie_features)
        movie_features_grad = -2/(numRated) * np.dot(difference.T, user_features)

        # cập nhật user_features và movie_features
        user_features -= learning_rate * user_features_grad
        movie_features -= learning_rate * movie_features_grad

        # tính giá trị hàm mất mát
        error = calc_lost(user_features, movie_features, ratings)
        if error < 1.6:
            print("iteration: ", i, "error: ", error)

        # chỉnh lại learning_rate
        if error < 1.6:
            learning_rate = 50
        elif error < 1.4:
            learning_rate = 30
        elif error < 1.2:
            learning_rate = 20
        elif error < 1:
            learning_rate = 10
        elif error < 0.5:
            learning_rate = 1

        # nếu error < 0.001 thì dừng quá trình gradient descent
        if error < 0.001:
            print("iteration: ", i, "error: ", error)
            break

    return user_features, movie_features

```

✓ 0.0s

```
# thiết lập các tham số cho gradient descent
learning_rate = 100
num_iterations = 100000

users_features, movie_features = gradient_descent(user_features, movie_features, ratings, learning_rate, num_iterations)

# in ra các ma trận đặc trưng của người dùng và movies
# print("user features: \n", user_features)
# print("\n")
# print("movie features: \n", movie_features)
```

✓ 20m 18.7s

```
iteration: 76 error: 1.5998954228981124
iteration: 77 error: 1.5998192696220994
iteration: 78 error: 1.5997433664694536
iteration: 79 error: 1.599667712242986
iteration: 80 error: 1.5995923057508226
iteration: 81 error: 1.5995171458063409
iteration: 82 error: 1.5994422312281291
iteration: 83 error: 1.5993675608399391
iteration: 84 error: 1.5992931334706568
iteration: 85 error: 1.5992189479542738
iteration: 86 error: 1.599145003129865
iteration: 87 error: 1.5990712978415662
iteration: 88 error: 1.5989978309385628
iteration: 89 error: 1.5989246012750704
iteration: 90 error: 1.5988516077103272
iteration: 91 error: 1.5987788491085793
iteration: 92 error: 1.5987063243390736
iteration: 93 error: 1.598634032276049
iteration: 94 error: 1.5985619717987267
iteration: 95 error: 1.598490141791303
iteration: 96 error: 1.5984185411429424
iteration: 97 error: 1.5983471687477702
iteration: 98 error: 1.5982760235048643
iteration: 99 error: 1.5982051043182481
iteration: 100 error: 1.5981344100968837
...
iteration: 99996 error: 1.5545012486497174
iteration: 99997 error: 1.5545012486497174
iteration: 99998 error: 1.5545012486497174
iteration: 99999 error: 1.5545012486497172
```

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...

- Thôi mệt quá kệ nó luôn nha :"))

## Tài liệu tham khảo

- [Machine Learning cơ bản \(machinelearningcoban.com\)](http://machinelearningcoban.com)
- [\(53\) Gradient Descent, Step-by-Step - YouTube](#)
- [4. Gradient Descent | Quy's blog \(ndquy.github.io\)](http://ndquy.github.io)
- [Machine Learning cơ bản \(machinelearningcoban.com\)](http://machinelearningcoban.com)
- [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)
- [Gradient Ascent: When to use it in machine learning? \(analyticsindiamag.com\)](http://analyticsindiamag.com)

- [What Is the Difference Between Gradient Descent and Gradient Ascent? | Baeldung on Computer Science](#)
- [machine learning - Gradient Ascent vs Gradient Descent in Logistic Regression - Cross Validated \(stackexchange.com\)](#)
- [Recommender Systems in Machine Learning: Examples - Data Analytics \(vitalflux.com\)](#)
- [Featured Hybrid Recommendation System Using Stochastic Gradient Descent | Atlantis Press \(atlantis-press.com\)](#)
- [\(71\) How does Netflix recommend movies? Matrix Factorization - YouTube](#)