**PHÂN TÍCH DỮ LIỆU L14**

**NHÓM 7:**

* **Huỳnh Phú Quang**
* **Tôn Nữ Hương Trân**
* **Nguyễn Thanh Long**

1. **Phân tích giá trị đơn lẻ và phân tích thành phần chính (20 điểm)**

Trong vấn đề này, chúng ta sẽ khám phá mối quan hệ giữa hai trong số các kỹ thuật giảm kích thước phổ biến nhất, SVD và PCA, ở mức khái niệm cơ bản. Trước khi chúng ta tiếp tục với câu hỏi, chúng ta hãy tóm tắt ngắn gọn các kỹ thuật SVD và PCA và một số nhận xét quan trọng:

* Đầu tiên, hãy nhớ lại rằng phép phân tích giá trị riêng của một ma trận thực, đối xứng và vuông B ( kích thước d x d) có thể được viết dưới dạng sản phẩm sau:

B=QΛQT

Trong đó Λ = diag(λ1, . . . , λd) chứa các giá trị riêng của B (luôn luôn là số thực) dọc theo đường chéo chính của nó và Q là một ma trận trực giao chứa các vectơ riêng của B dưới dạng các cột của nó.

* Principal Component Analysis (PCA): Cho trước một ma trận dữ liệu M ( kích thước p × q), PCA liên quan đến việc tính toán các vectơ riêng của MMT hoặc MT M. Ma trận của các vectơ riêng này có thể được coi là một phép quay cứng nhắc ở mức cao chiều không gian. Khi bạn áp dụng phép biến đổi này cho dữ liệu gốc, trục tương ứng với véc tơ riêng chính là trục dọc theo đó các điểm được "trải rộng" nhất. Chính xác hơn, trục này là trục mà phương sai của dữ liệu được cực đại hóa. Nói cách khác, các điểm tốt nhất có thể được xem là nằm dọc theo trục này, với độ lệch nhỏ so với trục này. Tương tự như vậy, trục tương ứng với vectơ riêng thứ hai (vector riêng tương ứng với giá trị riêng lớn thứ hai) là trục dọc theo đó phương sai của khoảng cách so với trục thứ nhất là lớn nhất, v.v.
* Singular Value Decomposition (SVD): SVD liên quan đến việc phân tách ma trận dữ liệu M ( kích thước p × g) thành một sản phẩm: UΣVT trong đó U ( kích thước p x k) và V ( kích thước q x k) là các ma trận trực giao cột và Σ ( kích thước k x k) là một ma trận đường chéo. Các mục dọc theo đường chéo của Σ được gọi là các giá trị đơn lẻ của M. Chìa khóa để hiểu những gì SVD cung cấp là xem các cột r của U, Σ và V dưới dạng biểu diễn các khái niệm ẩn trong ma trận gốc M.

Để trả lời các câu hỏi bên dưới, chúng ta hãy định nghĩa một ma trận thực M (có kích thước p × g) và giả sử ma trận này tương ứng với một tập dữ liệu có p điểm dữ liệu và q kích thước.

**(a)**

Ma trận MMT và MT M có đối xứng, vuông góc và thực không? Giải thích.

* GIẢI: Có, cả hai ma trận đều đối xứng, vuông và thực.
* Đối xứng: (MMT)T= MMT và (MTM)T= MTM
* Vuông: Khi bạn nhân một ma trận có kích thước p x q với phép hoán vị của nó có kích thước q x p, bạn sẽ có một ma trận vuông có kích thước p x p.
* Số thực: Cho rằng M là số thực, tích của M và chuyển vị của nó cũng sẽ là số thực.

**(b)**

Chứng minh rằng các trị riêng khác 0 của MMT giống với các trị riêng khác 0 của MTM. Bạn có thể bỏ qua bội số của các trị riêng. Vectơ riêng của chúng có giống nhau không?

* **GIẢI:** Cho M = UΣVT, trong đó U và V là các ma trận trực chuẩn cột duy nhất cho bởi SVD. Vì vậy:
* MTM= (UΣVT)T(UΣVT)

= (VΣTUT)(UΣVT) = VΣ2VT

Từ định nghĩa phân tách giá trị riêng, chúng ta thấy rằng MT M có các giá trị riêng khác 0 như các mục nhập đường chéo của Σ2

* MMT = (UΣVT)(UΣVT)T= (UΣVT)(VΣTUT) = UΣ2UT

Vì vậy, một lần nữa các giá trị riêng khác 0 của MMT là các mục nhập đường chéo của Σ2.

Từ hai giá trị trên, chúng tôi kết luận MTM và MMT có cùng các giá trị riêng khác 0.

Gọi e là véc tơ riêng của M MT và đặt giá trị riêng tương ứng là λi.e MMT(e) =λ(e). Nhân cả hai vế của phương trình với MT và kết quả là MT MMTe = MTλe có thể rút gọn thành MT M(MTe) = λ(MTe). Do đó, giá trị riêng của MT M = λ cũng vậy nhưng vectơ riêng là MTe khác với vectơ riêng của M MT là e.

**(c)**

Bây giờ chúng ta đã hiểu một số tính chất nhất định của MT M, hãy viết biểu thức cho MT M theo Q, QT và Λ trong đó Λ = diag(λ1, . . . , λd) chứa các giá trị riêng của MT M dọc theo đường chéo chính của nó và Q là ma trận trực giao chứa các vectơ riêng của MT M làm cột của nó?

**Gợi ý:** Kiểm tra định nghĩa về phân tách giá trị riêng được cung cấp ở đầu câu hỏi để xem liệu nó có thể áp dụng được không.

* **GIẢI:** MT M = QΛQT

**(d)**

SVD phân rã ma trận M thành tích UΣVT trong đó U và V là trực giao cột và Σ là ma trận chéo. Cho rằng M=UΣVT hãy viết biểu thức đơn giản hóa cho MT M theo V, VT và Σ.

* GIẢI: MT M = (UΣVT)TUΣVT

= VΣTUTUΣVT

= VΣTΣVT

= VΣ2VT

**(e)**

Trong câu hỏi này, chúng ta hãy kiểm tra bằng thực nghiệm xem phép phân tích SVD của M có thực sự cung cấp cho chúng ta các vectơ riêng (thứ nguyên PCA) của MT M hay không. Chúng tôi đặc biệt khuyến nghị sinh viên sử dụng Python và chức năng gợi ý cho bài tập này.? Khởi tạo ma trận M như sau:

M=

* Tính toán SVD của M (Sử dụng hàm *scipy.linalg.sod* trong Python và đặt đối số ma trận đầy đủ thành False). Hàm trả về các giá trị tương ứng với U, Σ và VT. Các giá trị được trả về cho U, Σ và VT là gì? Lưu ý: Đảm bảo rằng phần tử đầu tiên của mảng Σ được trả về có giá trị lớn hơn phần tử thứ hai.

*Những lần triển khai SVD và PCA sau này có thể cho kết quả hơi khác một chút. Ngoài ra, bạn sẽ chỉ cần ít hơn năm lệnh python để trả lời toàn bộ câu hỏi này.*

* Tính toán phân tách giá trị riêng của MT M (Sử dụng hàm *scipy.linalq.eigh* trong Python). Hàm trả về hai tham số: một danh sách các giá trị riêng (hãy gọi danh sách này là Evals) và một ma trận có các cột tương ứng với các vectơ riêng của các giá trị riêng tương ứng (hãy gọi ma trận này là Eves). Sắp xếp danh sách Evals theo thứ tự giảm dần sao cho giá trị riêng lớn nhất xuất hiện đầu tiên trong danh sách. Ngoài ra, hãy sắp xếp lại các cột trong Evecs sao cho vectơ cigen tương ứng với giá trị riêng lớn nhất xuất hiện trong cột đầu tiên của Evecs. Các giá trị của Evals và Eves (sau quá trình sắp xếp và sắp xếp lại) là gì?
* **Giải:** Một câu trả lời được chấp nhận là

>>>U

array( [ [ 0.27854301, 0.5 ],

[ 0.27854301, -0.5 ],

[ 0.64993368, 0.5 ],

[ 0.64993368, -0.5 ]]),

>>> S

array( [ 7.61577311, 1.41421356])

>>> Vh

array( [ [ 0.70710678, 0.70710678],

[ -0.70710678, 0.70710678] ] ）

>>> P

array( [ 2., 58.] )

>>> Q

array( [ [ -0.70710678, 0.70710678],

[ 0.70710678, 0.70710678] ] ）

**Câu trả lời khác được chấp nhận cho phần e là**

>>> import numpy

>>> from scipy import linalg

>>> M = numpy.array([[1, 2], [2, 1], [3, 4], [4, 3]])

>>> linalg.svd(M, full matrices=False)

(array([[-0.27854301, 0.5 ],

[-0.27854301, -0.5 ],

[-0.64993368, 0.5 ],

[-0.64993368, -0.5 ]]),

array([ 7.61577311, 1.41421356]),

array([[-0.70710678, -0.70710678],

[-0.70710678, 0.70710678]]))

>>> linalg.eigh(numpy.dot(numpy.transpose(M), M))

(array([ 2., 58.]), array([[-0.70710678, 0.70710678],

[ 0.70710678, 0.70710678]]))

* Dựa vào thí nghiệm và các biểu thức thu được ở phần (c) và phần (d) cho MT M, mối quan hệ (nếu có) giữa các giá trị riêng của MT M và các giá trị riêng của M? Giải thích.

Lưu ý: Các mục dọc theo đường chéo của Σ (phần (e)) được gọi là các giá trị riêng của M. Các giá trị riêng của MT M được bắt bởi các phần tử đường chéo trong Λ (phần (d))

* **GIẢI**: Các giá trị của M là căn bậc hai của các giá trị riêng của MT M

**Nội dung cần nộp:**

(i) Lời giải bằng văn bản cho các câu hỏi từ 1(a) đến 1( e) với các giải thích bất cứ nơi nào được yêu cầu

(ii) Tải mã lên qua Gradescope [1(e)]

**2. K-means on Spark (20 điểm):**

*Lưu ý*: Vấn đề này nên được triển khai trong Spark. Bạn không nên sử dụng thư viện Spark MLlibclustering cho vấn đề này. Bạn có thể lưu trữ các trọng tâm trong bộ nhớ nếu bạn chọn làm như vậy.

Vấn đề này sẽ giúp bạn hiểu các chi tiết cơ bản của việc triển khai các thuật toán phân cụm trên Spark. Ngoài ra, vấn đề này cũng sẽ giúp bạn hiểu tác động của việc sử dụng các số liệu khoảng cách và chiến lược khởi tạo khác nhau trong thực tế. Giả sử chúng ta có một tập hợp các điểm dữ liệu của X trong không gian d chiều Rd. Cho trước số lượng cụm và tập hợp k trọng tâm C, bây giờ chúng ta tiến hành xác định các số liệu khoảng cách khác nhau và các hàm chi phí tương ứng mà chúng tối thiểu hóa.

Khoảng cách Euclide Cho hai điểm A và B trong không gian có d chiều sao cho A=[a1, a2 ad] và B= [b1, b2 bd], khoảng cách Euclide giữa A và B được định nghĩa là: **(1)**

Hàm chi phí tương ứng φ được giảm thiểu khi chúng ta gán điểm cho các cụm bằng cách sử dụng thước đo khoảng cách Euclide được cho bởi: **(2)**

Lưu ý rằng trong hàm chi phí, giá trị khoảng cách được bình phương. Điều này là có chủ ý, vì nó là khoảng cách Euclide bình phương mà thuật toán được đảm bảo giảm thiểu.

Khoảng cách Manhattan Cho hai điểm ngẫu nhiên A và B trong không gian d chiều sao cho A= [a1, a2 ad] và B= [b1, b2 bd], khoảng cách Manhattan giữa A và B được định nghĩa là: **(3)**

Hàm chi phí tương ứng ψ được giảm thiểu khi chúng ta gán điểm cho các cụm bằng cách sử dụng số liệu khoảng cách Manhattan được đưa ra bởi: **(4)**

Thuật toán k-Means lặp: Chúng ta đã học thuật toán k-Means cơ bản trong lớp như sau: k trọng tâm được khởi tạo, mỗi điểm được gán cho trọng tâm gần nhất và các trọng tâm được tính toán lại dựa trên việc gán điểm cho các cụm. Trong thực tế, các bước trên được thực hiện trong một vài lần lặp lại. Chúng tôi trình bày phiên bản lặp kết quả của k-Means trong Thuật toán 1.

*Thuật toán 1*: Lặp k-Means Thuật toán

1: Quy trình k-Means lặp

2: Chọn k điểm làm trọng tâm ban đầu của k cụm.

3: Đối với các lần lặp := 1 đến MAX ITER thực hiện

4: Đối với mỗi điểm p trong tập dữ liệu thực hiện

5: Gán điểm p cho cụm có trọng tâm gần nhất

6: Kết thúc cho (end for)

7: Tính chi phí cho lần lặp này.

8: Đối với mỗi cụm c thực hiện

9: Tính lại trọng tâm của c là gtrị trung bình của tất cả các điểm dữ liệu được gán cho c

10: Kết thúc cho (end for)

11: Kết thúc cho (end for)

12: Kết thúc quy trình (end procedure)

K-means lặp đi lặp lại trên Spark: sử dụng K-means lặp đi lặp lại của Spark. sử dụng bộ dữ liệu từ q2/ data để giải quyết vấn đề này.

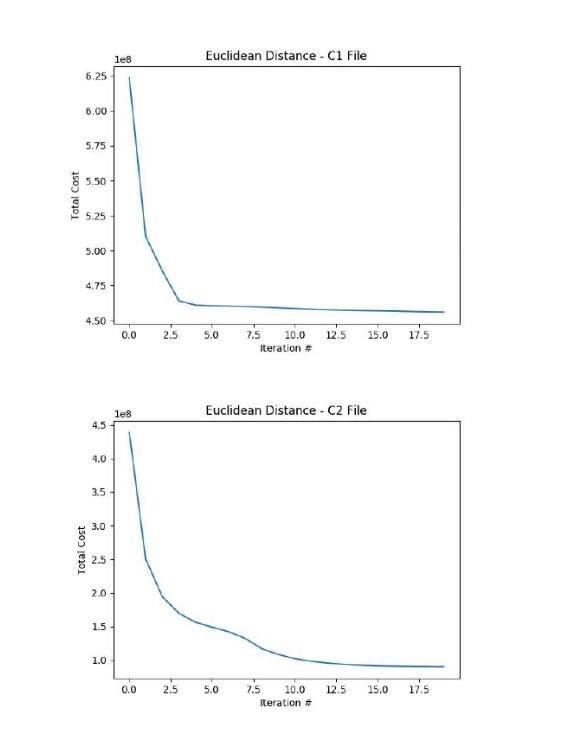
*Thư mục có 3 tệp:*

1. data.txt chứa tập dữ liệu có 4601 hàng và 58 cột. Mỗi hàng là một tài liệu được biểu diễn dưới dạng một vectơ đặc trưng 58 d. Mỗi thành phần trong vectơ thể hiện tầm quan trọng của một từ trong tài liệu. ID để downloaddata.txt vào Colab là 1E-voIV2ctU4Brw022Na8RHVVRGOoNkO1.
2. c1.txt chứa k trọng tâm cụm ban đầu. Các trọng tâm này đã được chọn bằng cách chọn k = 10 điểm ngẫu nhiên từ dữ liệu đầu vào. ID để tải c1.txt xuống Colabis 1yXNlZWMqUcAwDScBrkFChOHJwR1FZXmI.
3. c2.txt chứa trọng tâm cụm ban đầu cách xa nhau nhất có thể, sử dụng khoảng cách Euclide làm thước đo khoảng cách. (Bạn có thể làm điều này bằng cách chọn ngẫu nhiên trọng tâm thứ nhất c1, sau đó tìm điểm c2 cách xa c1 nhất, sau đó chọn c3 cách xa c1 và c2 nhất, v.v.). ID để tải c2.txt xuống Colabis 1vfovle9DgaeK0LnbQTH0j7kRaJjsvLt.

Đặt số lần lặp lại (MAX ITER) thành 20 và số cụm k thành 10 cho tất cả các thử nghiệm cũ được thực hiện trong câu hỏi này. Chương trình trình điều khiển của bạn phải đảm bảo rằng số lần lặp chính xác được chạy.

1. **Khám phá các chiến lược khởi tạo với khoảng cách Euclide [10 điểm]**
2. [5 điểm] Sử dụng khoảng cách Euclide (tham khảo Công thức 1) làm thước đo khoảng cách, tính hàm chi phí φ(i) (tham khảo Công thức 2) cho mỗi lần lặp i. Điều này có nghĩa là, đối với lần lặp đầu tiên của bạn, bạn sẽ tính toán hàm chi phí bằng cách sử dụng trọng tâm ban đầu nằm ở một trong hai tệp văn bản. Chạy k-mean trên data.txt bằng cách sử dụng c1.txt và c2.txt. Tạo biểu đồ trong đó bạn vẽ đồ thị hàm chi phí φ(i) dưới dạng hàm số lần lặp i=1..20 cho c1.txt và cả cho c2.txt. Bạn có thể sử dụng một biểu đồ hoặc hai biểu đồ khác nhau, bất kỳ biểu đồ nào bạn cho là tốt nhất để trả lời các câu hỏi lý thuyết mà chúng tôi đang hỏi bạn.

*(Gợi ý: Lưu ý rằng bạn không cần phải viết một công việc Spark riêng để tính toán φ(i). Bạn có thể tính toán chi phí trong khi phân vùng các điểm thành các cụm.)*

**GIẢI: Chi phí so với Lặp lại:**

1. [5 điểm] Phần trăm thay đổi về chi phí sau 10 lần lặp lại K-Meansalgorithm khi các trọng tâm của cụm được khởi tạo bằng cách sử dụng c1.txt so với c1.txt. c2.txt và chỉ số khoảng cách đang được sử dụng là khoảng cách Euclide? Khởi tạo ngẫu nhiên k-means bằng c1.txt có tốt hơn khởi tạo bằng c2.txt xét về chi phí φ(i) không? Giải thích lý do của bạn.

(Gợi ý: để rõ ràng, tỷ lệ phần trăm đề cập đến (chi phí[0]-chi phí[10])/chi phí[0].)

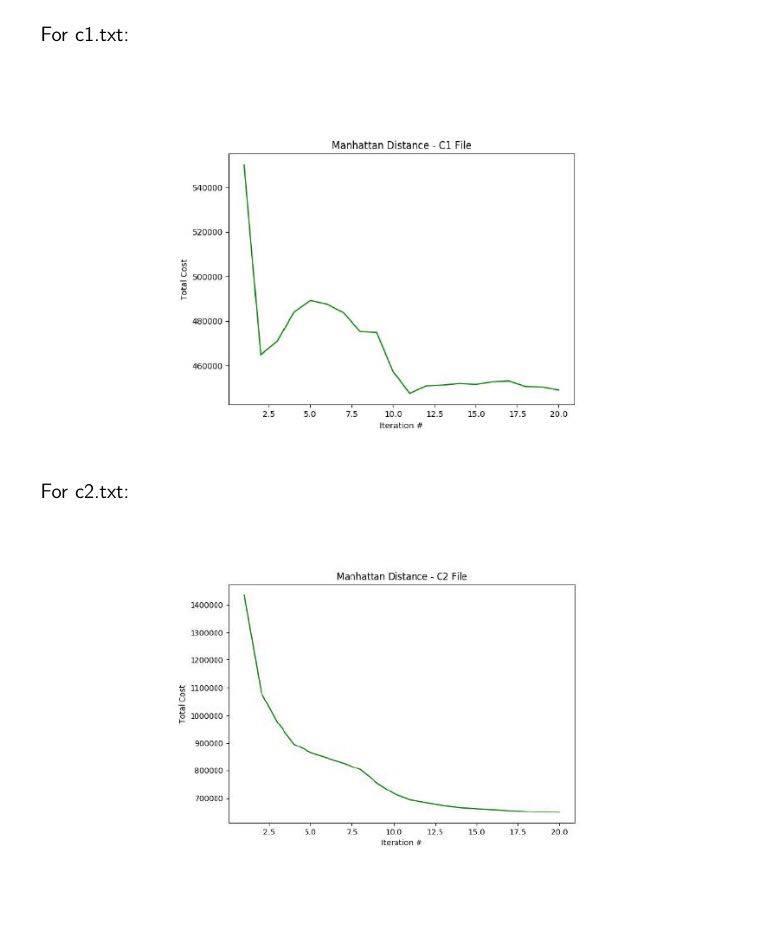
**GIẢI:**

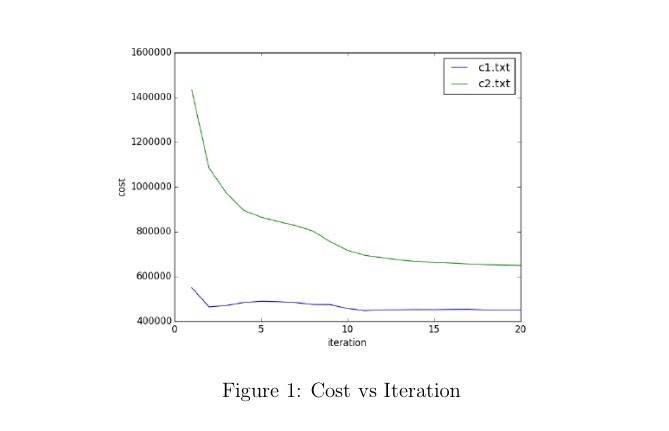
* c1 cải thiện 26% sau 10 lần lặp.
* c2 cải thiện 75% sau 10 lần lặp.
* c2 tốt hơn c1 vì nó phân phối các cụm ban đầu cách xa nhau. Bởi vì có ít chồng chéo hơn, các cụm thực sự được phân chia ít thường xuyên hơn, dẫn đến tập hợp các cụm cuối cùng tốt hơn.

1. **Khám phá các chiến lược khởi tạo với khoảng cách Manhattan [10 điểm]**
2. [5 điểm] Sử dụng thước đo khoảng cách Manhattan (tham khảo Công thức 3) làm thước đo khoảng cách, tính hàm chi phí ψ(i) (tham khảo Phương trình 4) cho mỗi lần lặp i. Điều này có nghĩa là, đối với lần lặp đầu tiên của bạn, bạn sẽ tính toán hàm chi phí bằng cách sử dụng trọng tâm ban đầu nằm ở một trong hai tệp văn bản. Chạy k-mean trên data.txt bằng cách sử dụng c1.txt và c2.txt. Tạo biểu đồ trong đó bạn vẽ biểu đồ hàm chi phí ψ(i) dưới dạng hàm số lần lặp i=1..20 cho c1.txt và cả cho c2.txt. Bạn có thể sử dụng một biểu đồ hoặc hai biểu đồ khác nhau, bất kỳ biểu đồ nào bạn cho là tốt nhất để trả lời các câu hỏi lý thuyết mà chúng tôi đang hỏi bạn.

*(Gợi ý: Bài toán này có thể giải tương tự như phần (a). Cũng lưu ý rằng có thể đối với khoảng cách Manhattan, chi phí không phải lúc nào cũng giảm. K-mean chỉ đảm bảo giảm chi phí đơn điệu c cho Euclidean bình phương khoảng cách. Tra cứu K-trung vị để tìm hiểu thêm.)*

**GIẢI : Chi phí so với Lặp lại:**





1. [5 điểm] Phần trăm thay đổi về chi phí sau 10 lần lặp lại K-Meansalgorithm khi các trọng tâm của cụm được khởi tạo bằng cách sử dụng c1.txt so với c1.txt. c2.txt và chỉ số khoảng cách đang được sử dụng là khoảng cách Manhattan? Khởi tạo ngẫu nhiên k-means bằng c1.txt có tốt hơn khởi tạo bằng c2.txt xét về chi phí ψ(i) không? Giải thích lý do của bạn.

**GIẢI:**

* c1 cải thiện 19%.
* c2 cải thiện 52%.
* c2 không tốt hơn c1 đối với khoảng cách Manhattan vì các điểm trong c2.txt cách xa nhau nhất có thể khi sử dụng chỉ số khoảng cách Euclide và không không nhất thiết phải xa nhau ở khoảng cách Manhattan

**Những gì cần gửi:**

1. Tải mã cho 2(a) và 2(b) lên Gradescope
2. Biểu đồ chi phí so với lặp lại cho hai chiến lược khởi tạo [2(a)]
3. Giá trị phần trăm cải thiện và giải thích của bạn [2(a)]
4. Sơ đồ chi phí so với giá trị lặp lại cho hai chiến lược khởi tạo [2(b)]
5. Giá trị phần trăm cải thiện và giải thích của bạn [2(b)]
6. **Latent Features for Recommendations (35 điểm):**

*Lưu ý*: Vui lòng sử dụng Python gốc (không yêu cầu Spark) để giải quyết vấn đề này. Quá trình này thường mất vài phút để chạy, tuy nhiên, thời gian có thể khác nhau tùy thuộc vào hệ thống bạn sử dụng.

Mục tiêu của vấn đề này là triển khai thuật toán Stochastic Gradient Descent để xây dựng hệ thống Đề xuất Yếu tố tiềm ẩn. Chúng tôi có thể sử dụng nó để giới thiệu phim cho người dùng. Chúng tôi khuyến khích bạn đọc lại các slide của bài giảng “Recommender Systems 2” trước khi thử giải bài toán.

Giả sử chúng ta được cung cấp một ma trận xếp hạng R. Phần tử Riu của ma trận này tương ứng với xếp hạng do người dùng u đưa ra cho mục i. Kích thước của R là m×n, trong đó m là số lượng phim và n là số lượng người dùng.

Hầu hết các yếu tố của ma trận đều không xác định vì mỗi người dùng chỉ có thể xếp hạng một số phim. Mục tiêu của chúng ta là tìm hai ma trận P và Q sao cho R≃QPT. Kích thước của Q là m×k và kích thước của P là n×k, k là các tham số của thuật toán.

Chúng tôi xác định lỗi là: **(5)**

có nghĩa là chúng tôi chỉ tính tổng trên các cặp (người dùng, mục) mà người dùng đã xếp hạng mục đó, tức là mục (i, u) của ma trận R là đã biết. qi biểu thị hàng thứ i của ma trận Q (tương ứng với một mục) và pu là hàng thứ u của ma trận P (tương ứng với người dùng u). qi và pu đều là các vectơ hàng có kích thước k. λ là tham số chuẩn hóa. là chuẩn L2 và là bình phương của chuẩn L2, tức là tổng bình phương các phần tử của pu .

(a) [10 điểm]Gọi ε iu là đạo hàm của sai số E đối với Riu. biểu thức cho εiu là gì? Các phương trình cập nhật cho qi và pu trong thuật toán Giảm dốc ngẫu nhiên là gì? Vui lòng chỉ ra nguồn gốc và sử dụng εiu trong biểu thức cuối cùng của bạn về qi và pu.

**GIẢI:**

1. [25 điểm] Thực hiện thuật toán. Đọc từng mục của ma trận R từ đĩa và cập nhật εiu, qi và pu cho từng mục. Để nhấn mạnh, bạn không được phép lưu trữ bộ nhớ R trong ma trận. Bạn phải đọc từng phần tử Riu một lần từ đĩa và áp dụng các phương trình đã cập nhật của bạn (cho từng phần tử) mỗi lần lặp. Mỗi lần lặp lại của thuật toán sẽ đọc toàn bộ tệp.

Chọn k= 20, λ= 0.1 và số lần lặp = 40. Tìm giá trị tốt cho tốc độ học η, bắt đầu với η= 0,1. (Bạn không thể sửa đổi k hoặc λ) Lỗi E trên tập huấn luyện setratings.train.txt được thảo luận bên dưới phải nhỏ hơn 65000 sau 40 lần lặp lại; bạn nên quan sát cả qi và pu ngừng thay đổi.

Dựa vào giá trị của η, bạn có thể gặp các trường hợp sau:

• Nếu η quá lớn, hàm sai số có thể hội tụ về giá trị cao hoặc có thể giảm không đơn điệu. Nó thậm chí có thể phân kỳ và làm cho các thành phần của vectơ p và q bằng ∞. • Nếu η quá nhỏ, hàm sai số không có thời gian để giảm đáng kể và đạt đến sự hội tụ. Vì vậy, nó có thể đơn điệu giảm nhưng không hội tụ, tức là nó có thể có giá trị cao sau 40 lần lặp vì nó chưa hội tụ.

Sử dụng tập dữ liệu tại q3/data trong gói cho vấn đề này. Nó chứa các tệp sau: •ratings.train.txt: Đây là trận đấu R. Mỗi mục nhập bao gồm id người dùng, id phim và xếp hạng.

Vẽ giá trị của hàm mục tiêu E (được xác định trong phương trình 5) trên tập huấn luyện dưới dạng một hàm của số lần lặp lại. Bạn đã tìm thấy giá trị nào của η?

Bạn có thể sử dụng bất kỳ ngôn ngữ lập trình nào để triển khai phần này, nhưng Java, C/C++ và Python được khuyên dùng để tăng tốc độ. (Đặc biệt, Matlab có thể đọc từ đĩa khá chậm.) Có thể có một giải pháp mất vài phút để chạy với các ngôn ngữ này.

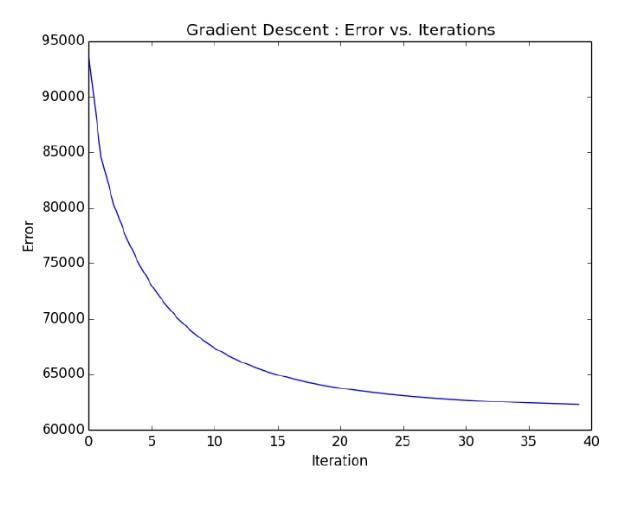
*Gợi ý*: Những gợi ý này sẽ giúp ích cho bạn nếu bạn không chắc chắn về cách tiến hành các bước nhất định của thuật toán, mặc dù bạn không cần phải làm theo chúng nếu bạn có một phương pháp khác.

•Khởi tạo P và Q: Chúng tôi muốn qi và pu cho tất cả người dùng u và các mục i sao cho . Một cách hay để đạt được điều đó là khởi tạo tất cả các phần tử của P và Q thành các giá trị ngẫu nhiên trong .

•Cập nhật các phương trình: Trong mỗi lần cập nhật, chúng tôi cập nhật qi bằng pu và pu bằng qi. Tính toán các giá trị mới cho qi và pu bằng cách sử dụng các giá trị cũ, sau đó cập nhật các vectơ qi và pu.

• Bạn nên tính E khi kết thúc toàn bộ quá trình đào tạo. Tính toán E thành từng phần trong quá trình lặp lại là không chính xác vì P và Q vẫn đang được cập nhật.

**GIẢI**: **η= 0,03**



**Những gì cần gửi:**

1. Ptrinh cho ε iu. Cập nhật các phương trình trong thuật toán Giảm dốc ngẫu nhiên [3(a)]
2. (Giá trị của η. Số lần lặp lại. Đảm bảo biểu đồ của bạn có trục y để chúng tôi có thể đọc giá trị của E. Chỉ cần một biểu đồ với η bạn đã chọn [3(b)]
3. Vui lòng tải tất cả mã lên Gradescope [3(b)]

**4 Hệ thống khuyến nghị**

**Lưu ý:** Vui lòng sử dụng Python gốc (không yêu cầu Spark) để giải quyết vấn đề này. Nếu bạn gặp phải lỗi bộ nhớ khi thực hiện các phép toán ma trận lớn, vui lòng đảm bảo rằng bạn đang sử dụng Python 64 bit thay vì 32 bit (có giới hạn bộ nhớ 4GB).

Xem xét biểu đồ hai bên mục user-item trong đó mỗi cạnh trong biểu đồ giữa user U đến mục I, chỉ ra rằng user U thích mục I. Chúng tôi cũng đại diện cho ma trận xếp hạng cho nhóm user này và các mục là R, trong đó mỗi hàng trong R tương ứng với một user và mỗi cột tương ứng với một item. Nếu user i thích item j, thì Ri,j = 1, ngược lại Ri,j = 0. Cũng giả sử chúng ta có m user và n mục, do đó ma trận R là m × n.

Hãy định nghĩa ma trận P, m × m, là một ma trận đường chéo có phần tử đường chéo i-th thứ i là

mức độ của nút user i, tức là số lượng mục mà user thích. Tương tự, một ma trận Q,n×n là một ma trận đường chéo có phần tử đường chéo thứ i là bậc của nút item i hoặc số

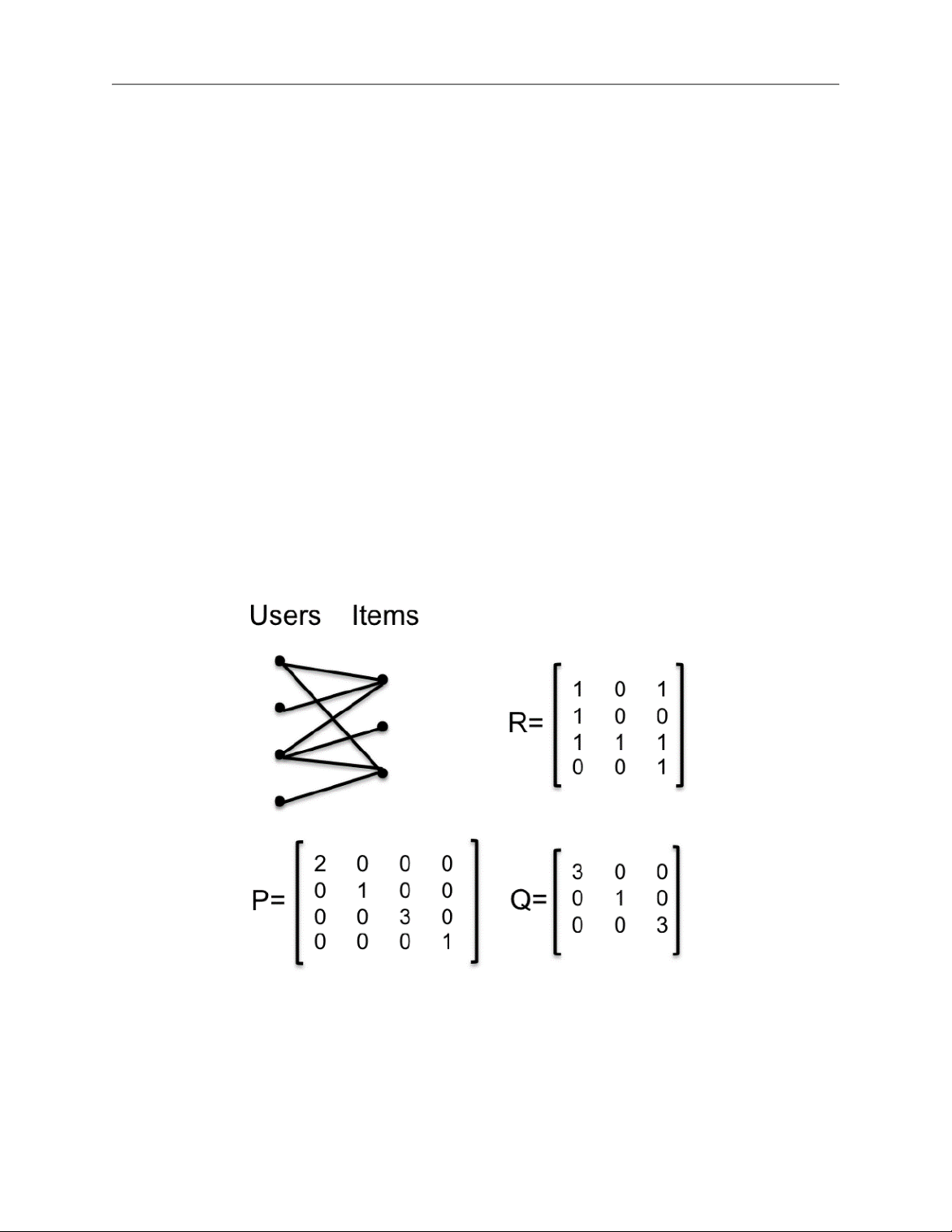
của user đã thích item i.

(a)

Xác định ma trận tương tự user không chuẩn hóa T = R ∗ RT(phép nhân của Rand

chuyển vị R). Giải thích ý nghĩa của Tii and Tij ( i j ), dưới dạng đồ thị hai phía

cấu trúc (ví dụ: độ nút, đường dẫn giữa các nút, v.v.)



BÀI LÀM : Trong biểu đồ hai phía user-item, Tii bằng cấp của user. Vì Tii = = Rij ∗ (RT)ij = = Rij. Vì Rij là 0 hoặc 1 nên Tii = độ (useri).

Tij(i j) là số đường dẫn giữa useri and userj với độ dài là 2, nó cũng đại diện cho số item mà cả hai cùng thích. Tij = Rik ∗ Rjk, Rik ∗Rjk(0) nghĩa là tồn tại một đường dẫn 2 bước bắt đầu từ useri đến userj qua itemk. **Cosine Similarity:** độ tương tự cosine của hai vectơ u và v được định nghĩa là:

cos-sim(u,v) =

(b)

Hãy xác định ma trận độ tương tự item, SI,n × n, sao cho phần tử trong hàng i và cột là độ đồng dạng cosine của mục i và mục j tương ứng với cột i và cột j của ma trận R. Chứng tỏ rằng SI=Q−1/2RTRQ−1/2, trong đó Q−1/2 được xác định bởi = 1 cho tất cả các mục nhập khác 0 của ma trận và 0 ở tất cả các vị trí khác. Lặp lại câu hỏi tương tự cho ma trận độ tương tự của user, SU trong đó phần tử trong hàng i và cột là độ tương tự cosin của user i và user j tương ứng với hàng i và hàng j của ma trận R . Nghĩa là, biểu thức của bạn cho SU cũng phải dưới dạng một tổ hợp nào đó của R, P và Q. Câu trả lời của bạn phải là một phép toán trên ma trận, cụ thể là bạn không nên xác định từng hệ số của SU một cách riêng lẻ. biểu thức. (Lưu ý: Để tạo căn bậc hai theo từng phần tử của ma trận, bạn có thể viết nó dưới dạng ma trận lũy thừa của .)

BÀI LÀM: Chúng ta biểu thị bằng Ri. hàng thứ i của R, and by R·j là cột thứ j của R.

Chúng ta biết rằng vectơ của một phần tử được xác định bởi một cột là R. Hơn nữa, chuẩn của , giả sử, itemiis được xác định bởi . rên thực tế, tổng là số user thích mục này (vì Rki có thể là 0 hoặc 1, chúng ta có Rki =). Vậy chuẩn của itemi trên thực tế là .

Do đó, ta có:

cos-sim(itemi, itemj) =

=

Bây giờ, ma trận Q là đường chéo và các hệ số đường chéo của nó là tất cả khác 0 (nếu không, một số item không được ai thích, và chúng ta cũng có thể loại bỏ chúng, vì chúng vô dụng và vì góc mà chúng tạo với các item khác sẽ không được xác định), vì vậy chúng ta có thể biểu thị bằng Q⋆ ma trận đường chéo có hệ số đường chéo được xác định bởi . Sau đó chúng ta có điều đó:

(Q⋆RTRQ⋆)ij =

= ( Bởi vì Q⋆ là đường chéo)

=

=

Vì vậy, ma trận SI có thể được biểu diễn dưới dạng Q và R:

SI = Q⋆RTRQ⋆

Để tính toán một biểu thức tương tự cho Su, chúng tôi nhận thấy rằng (R, Q, SI) và (RT, P, Su) đóng vai trò tương tự nhau. Thật vậy, mối quan hệ "useru thích itemi" có thể được đặt ngược lại thành "itemi được thích bởi useru ", tương đương với việc chuyển đổi users và items, tức là chuyển vị ma trận R. Do đó, Su đã đưa ra bởi:

Su = P⋆RRTP⋆

với P⋆ là ma trận đường chéo có các hệ số được xác định bởi

(c) Phương pháp đề xuất sử dụng bộ lọc cộng tác người dùng-người dùng cho người dùng u, có thể được mô tả như sau: đối với tất cả các mục s, tính ru,s = cos-sim(x, u) ∗ Rxs và đề xuất các bộ diều mà ru,s là lớn nhất.

Tương tự, phương pháp đề xuất sử dụng lọc cộng tác mục-mục cho người dùng u có thể được mô tả như sau: đối với tất cả các mục s, tính ru,s = Rux ∗ cos-sim(x, s) và đề xuất k items cho ru,s  nào là lớn nhất.

Hãy xác định ma trận đề xuất, Γ, m×n, sao cho Γ(i, j) = ri,j. Tìm Γ cho cả phương pháp lọc cộng tác giữa item-item và user-user, theo R,P và Q. Câu trả lời cuối cùng của bạn nên mô tả các hoạt động ở cấp độ ma trận, chứ không phải các thuật ngữ cụ thể của ma trận.

Gợi ý: Đối với trường hợp mục-mục, Γ = RQ-1/2RTRQ−1/2.

Câu trả lời của bạn sẽ cho biết cách bạn rút ra các biểu thức (ngay cả đối với trường hợp item-item, nơi chúng tôi cung cấp cho bạn biểu thức cuối cùng).

BÀI LÀM: Đối với đề xuất lọc cộng tác user-user , chúng tôi có:

Γij = rij

=cos-sim(x, i) Rx,j

=cos-sim(i, x) Rx,j

=i,x R(x, j)

=(SuR)ij

=(P⋆RRTP⋆R)ij

Vì vậy,

Γ = P⋆RRTP⋆R

Tương tự, Đối với đề xuất lọc cộng tác item-item , chúng tôi có:

Γij = rij

=Ri,x cos-sim(x, i)

=(SI)x,j

=(RSI)ij

=(RQ⋆RTRQ⋆)ij

Thus,

Γ = RQ⋆RTRQ⋆