MỘT SỐ BÀI TOÁN ỨNG DỤNG

Nguyễn Mạnh Hùng

ĐẠI HỌC GTVT

03 - 2023

Nội dung

Hệ thống gợi ý

- Hệ thống gợi ý (Recommender System - RS) là một dạng hệ thống lọc thông tin tìm kiếm dự báo "*đánh giá*" hoặc "*sở thích*" của người dùng (user) với một sản phẩm hoặc đối tượng nào đó (item).
- Các hệ thống gợi ý được ứng dụng trong nhiều lĩnh vực:
  1. Netflix, YouTube và Spotify sẽ nhận diện sở thích của người dùng và gợi ý danh sách phát nhạc, dựa trên mức độ tương tác của người dùng với một bản nhạc nào đó.
  2. Amazon đưa ra gợi ý mua sắm dựa trên tần suất tìm kiếm sản phẩm và lịch sử mua hàng của người dùng.
  3. Facebook, Zalo đưa ra gợi ý kết bạn dựa trên các mối liên kết chung.
- Các hệ thống gợi ý được chia thành 2 nhóm chính:
  1. *Content based system*: đánh giá các thuộc tính của các *item* mà *user* "ưa thích", từ đó gợi ý các *item* có chung thuộc tính đó.
  2. *Collaborative filtering*: dựa trên sự tương quan giữa các *user* và/hoặc *item*, một nhóm *item* được gợi ý tới một *user* dựa trên các *user* có hành vi tương tự.

Ma trận utility

- Trong một hệ thống gợi ý, người ta quan tâm đến 3 thông tin chính là *item*, *user*, và *rating* (đánh giá) của *user* về *item*. Giá trị *rating* phản ánh mức độ quan tâm của *user* đối với *item*.
- Ma trận utility: biểu diễn các thông tin về *item*, *user*, và *rating*.

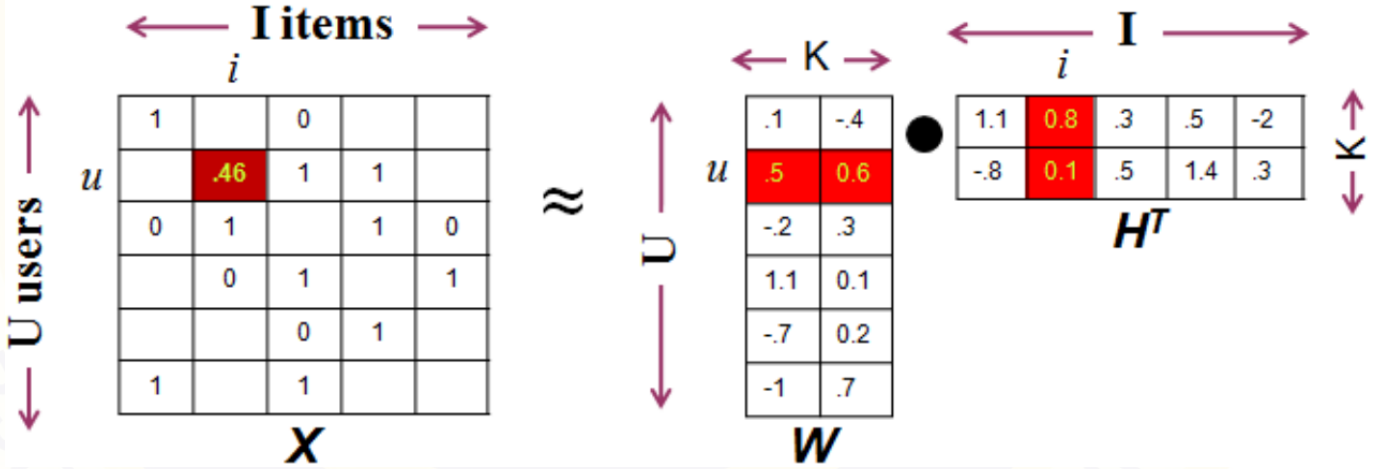
	item 1	*item* 2	*item* 3	*item* 4	*item* 5	*item* 6
user 1	4	5	1		2	1
user 2		4		3		2
user 3	3		1		3	
user 4	2			4		1
user 5	5		3	4	3	

- Các ô màu xanh không có đánh giá của *user* về *item*. Hệ thống gợi ý cần phải tự điền các giá trị này.

Bài toán phân tích ma trận

Kĩ thuật phân tích ma trận (Matrix Factorization - MF) có thể được sử dụng để điền các giá trị còn thiếu vào ma trận utility:

- *Phân tích ma trận utility* X *thành hai ma trận có kích thước nhỏ hơn là* W *và* H *sao cho tích ma trận* $W H^T$ *xấp xỉ* X *với độ chính xác cao:*



- *Ma trận* W *mô tả* K *thuộc tính ẩn (latent factor) của các user và ma trận* H *mô tả* K *thuộc tính ẩn của các item.*

Hàm mất mát (loss function)

- Kí hiệu $R = (r_{ui})$ là ma trận mô tả việc *user* đánh giá *item* hay không, $r_{ui} = 1$ nghĩa là *user u* có đánh giá *item i*, $r_{ui} = 0$ nếu ngược lại.
- Hai ma trận W và H được tìm bằng cách cực tiểu hóa hàm mất mát. Chẳng hạn, một dạng đơn giản của hàm mất mát như sau:

$$\mathcal{L}(W, H) = \sum_{(u,i):r_{ui}=1} \left(x_{ui} - \sum_{j=1}^{K} w_{uj} h_{ij} \right)^2$$

Chẳng hạn

$$\begin{pmatrix} 0.1 & -(0.4) \\ 0.5 & 0.6 \\ -(0.2) & 0.3 \\ 1.1 & 0.1 \\ -(0.7) & 0.2 \\ -1 & 0.7 \end{pmatrix} \cdot \begin{pmatrix} 1.1 & 0.8 & 0.3 & 0.5 & -2 \\ -(0.8) & 0.1 & 0.5 & 1.4 & 0.3 \end{pmatrix} = \begin{pmatrix} \boxed{0.43} & 0.04 & \boxed{-0.17} & -0.51 & -0.32 \\ 0.07 & 0.46 & \boxed{0.45} & \boxed{1.1} & -0.82 \\ \boxed{-0.46} & \boxed{-0.13} & 0.09 & \boxed{0.32} & \boxed{0.49} \\ 1.1 & \boxed{0.89} & \boxed{0.38} & 0.69 & \boxed{-2.2} \\ -0.93 & -0.54 & \boxed{-0.11} & \boxed{-0.07} & 1.5 \\ \boxed{-1.7} & -0.73 & \boxed{0.05} & 0.48 & 2.2 \end{pmatrix}$$

$$\mathcal{L}(W, H) = (1 - 0.43)^2 + (0 + 0.17)^2 + \cdots = 23.6233$$

Hiệu chỉnh hàm mất mát (Regularization)

Để ngăn chặn hiện tượng quá khớp (overfitting), người ta thay đổi hàm mất mát bằng cách thêm vào một đại lượng gọi là hiệu chỉnh (regularization) để điều khiển độ lớn của các giá trị trong W và H:

$$\mathcal{L}(\mathrm{W},\mathrm{H}) = \sum_{(u,i):r_{ui}=1} \left(x_{ui} - \sum_{j=1}^{K} w_{uj} h_{ij} \right)^2 + \lambda \left(\|\mathrm{W}\|_F^2 + \|\mathrm{H}\|_F^2 \right)$$

trong đó λ là hệ số chuẩn hóa và $\|A\|_F$ là chuẩn Frobenius của ma trận A,

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$$

Tìm nghiệm tối ưu

Để tìm nghiệm tối ưu cho W và H, ta lần lượt tối ưu một ma trận trong khi cố định ma trận còn lại cho tới khi hội tụ.

- Khi cố định W, ta tối ưu hóa H với hàm mất mát:

$$\mathcal{L}(\mathrm{H}) = \sum_{(u,i):r_{ui}=1} \left(x_{ui} - \sum_{j=1}^{K} w_{uj} h_{ij} \right)^2 + \lambda \|\mathrm{H}\|_F^2$$

- Khi cố định H, ta tối ưu hóa W với hàm mất mát:

$$\mathcal{L}(\mathrm{W}) = \sum_{(u,i):r_{ui}=1} \left(x_{ui} - \sum_{j=1}^{K} w_{uj} h_{ij} \right)^2 + \lambda \|\mathrm{W}\|_F^2$$

Tìm nghiệm tối ưu

Sử dụng thuật toán tối ưu **Gradient descent**, ta thu được công thức cập nhật các phần tử của hai ma trận W và H từ hệ phương trình như sau:

$$w_{uk}^{new} = w_{uk} + \beta \left[2 \left(x_{ui} - \sum_{j=1}^{K} w_{uj} h_{ij} \right) h_{ik} - \lambda w_{uk} \right]$$

$$h_{ik}^{new} = h_{ik} + \beta \left[2 \left(x_{ui} - \sum_{j=1}^{K} w_{uj} h_{ij} \right) w_{uk} - \lambda h_{ik} \right]$$

trong đó β là tốc độ học ($0 < \beta < 1$). Lúc ban đầu, hai ma trận W và H sẽ được khởi tạo ngẫu nhiên.

Phân tích ma trận - Ví dụ giải số

Bài toán: Phân tích ma trận utility

$$X = \begin{bmatrix} 5 & 3 & 0 & 1 \\ 4 & 0 & 0 & 1 \\ 1 & 1 & 0 & 5 \\ 1 & 0 & 0 & 4 \\ 0 & 1 & 5 & 4 \end{bmatrix}$$

thành tích $W H^T$, với $W \in \mathbb{R}^{5\times 2}$, $H \in \mathbb{R}^{4\times 2}$. Sử dụng hàm mất mát dạng:

$$\mathcal{L}(W, H) = \sum_{(u,i):x_{ui}\neq 0} \left(x_{ui} - \sum_{j=1}^{K} w_{uj} h_{ij} \right)^2$$

Ví dụ giải số

Xây dựng hàm thực hiện phân tích ma trận:

```python
import numpy as np
```

```
"""
@INPUT:
    A     : a matrix to be factorized, dimension M x N
    W     : an initial matrix of dimension M x K
    H     : an initial matrix of dimension N x K
    K     : the number of latent features
    steps : the maximum number of steps to perform the optimisation
    beta  : the learning rate
@OUTPUT:
    the final matrices W and H
"""
```

Ví dụ giải số

```
def matrix_factor(A, W, H, K, steps=5000, beta=0.0002):
    H = H.T
    for step in range(steps):
        for i in range(len(A)):
            for j in range(len(A[i])):
                if A[i][j] > 0:
                    eij = A[i][j] - np.dot(W[i,:],H[:,j])
                    for k in range(K):
                        W[i][k] = W[i][k] + beta*(2*eij*H[k][j])
                        H[k][j] = H[k][j] + beta*(2*eij*W[i][k])
        e = 0
        for i in range(len(A)):
            for j in range(len(A[i])):
                if A[i][j] > 0:
                    e = e + pow(A[i][j]-np.dot(W[i,:],H[:,j]), 2)
        if e < 0.0001:
            break
    return W, H.T
```

Ví dụ giải số

```
# Khởi tạo ma trận
A = np.array([[5,3,0,1],
              [4,0,0,1],
              [1,1,0,5],
              [1,0,0,4],
              [0,1,5,4]])

M = len(A)
N = len(A[0])
K = 2

W = np.random.rand(M,K)
H = np.random.rand(N,K)
```

Ví dụ giải số

```
nW, nH = matrix_factor(A, W, H, K)
print(nW)
print(nH)
np.dot(nW,nH.T)
```

```
[[0.6543035  2.30374006]
 [0.60303339 1.80998947]
 [2.11695809 0.24661478]
 [1.69575326 0.23855388]
 [1.77211928 0.7276636 ]]
[[ 0.26725979  2.10564034]
 [ 0.22926983  1.20501075]
 [ 2.06948627  1.70549746]
 [ 2.38631769 -0.24609887]]
```

```
A = np.array([[5,3,0,1],
              [4,0,0,1],
              [1,1,0,5],
              [1,0,0,4],
              [0,1,5,4]])
```

```
array([[5.02571703, 2.92604359, 5.28309495, 0.99442821],
       [3.97235342, 2.31931413, 4.33490177, 0.99359289],
       [1.0850598 , 0.78252809, 4.8016166 , 4.99104293],
       [0.95551534, 0.67624506, 3.91619114, 3.98789817],
       [2.00581405, 1.28313595, 4.90840494, 4.0497624 ]])
```

Bài toán hồi quy (Regression)

- Nhiều bài toán trong các lĩnh vực khoa học, kĩ thuật liên quan đến việc xác định quan hệ giữa các biến, chẳng hạn:
  1. Thu nhập y của một kĩ sư phụ thuộc vào trình độ học vấn x_1 và số năm kinh nghiệm x_2.
  2. Giá nhà y phụ thuộc vào diện tích x_1, số phòng ngủ x_2, và khoảng cách đến khu vực trung tâm x_3.
  3. Cường độ chịu nén của đất y phụ thuộc vào địa điểm khảo sát (x_1, x_2) và độ sâu x_3 tính từ mặt đất.
- Xây dựng một mô hình mô tả mối quan hệ giữa biến phụ thuộc y và một hay nhiều biến độc lập $x_1, x_2, \ldots, x_k$:

$$y_n = f(\mathrm{x_n}; \beta) + \epsilon_n$$

với $\mathrm{x}_n = (x_{n1}, x_{n2}, \ldots, x_{nk})^T$, y_n là giá trị của các biến tại quan sát thứ n, $\epsilon_n \sim N(0, \sigma^2)$ mô tả sai số ngẫu nhiên, và β là véc tơ tham số của mô hình cần xác định.

Mô hình hồi quy tuyến tính (Linear Regression Model)

- Mô hình đơn giản nhất là mô hình hồi quy tuyến tính:

$$y_n = \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \cdots + \beta_k x_{nk} + \epsilon_n = \mathrm{x}_n^T \beta + \epsilon_n$$

- Tập dữ liệu quan sát: $\{y_n, x_{n1}, x_{n2}, \ldots, x_{nk}\}$ với $n = 1, 2, \ldots, N$
- Mô hình cho n quan sát có thể được viết lại dưới dạng sau:

$$\mathrm{y} = \mathrm{X}\beta + \epsilon$$

trong đó $\mathrm{y} = (y_1, y_2, \ldots, y_N)^T$, $\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_N)^T$ và

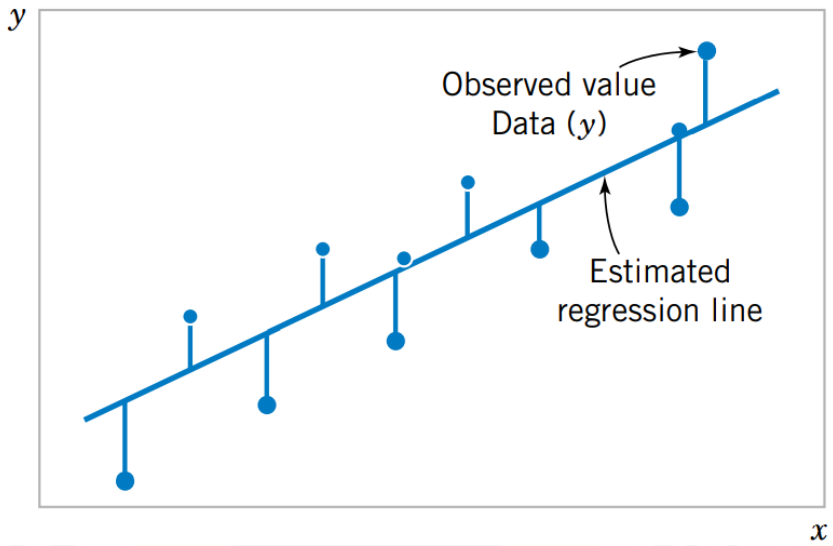
$$\mathrm{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Nk} \end{pmatrix}$$

Mô hình hồi quy tuyến tính

- Rõ ràng ta không thể xác định chính xác véc tơ tham số β của mô hình. Do đó, ta tìm một ước lượng B của nó. Khi đó, giá trị quan sát y_n có thể được xấp xỉ bởi mô hình:

$$y_n \approx \mathrm{x}_n^T \mathrm{B}$$

- Sai số dự báo của mô hình: $e_n = y_n - \mathrm{x}_n^T \mathrm{B}$



Mô hình hồi quy tuyến tính

- Ước lượng B làm cực tiểu hóa hàm tổng bình phương sai số:

$$SSE = \|\mathrm{y} - \mathrm{X}\,\mathrm{B}\|^2 = \sum_{n=1}^{N} \left(y_n - \mathrm{x}_n^T\,\mathrm{B}\right)^2$$

- Khi đó, ước lượng B là nghiệm của hệ phương trình:

$$(\mathrm{X}^T\mathrm{X})\,\mathrm{B} = \mathrm{X}^T\mathrm{y}$$

Nếu ma trận $(\mathrm{X}^T\mathrm{X})$ khả nghịch thì $\mathrm{B} = (\mathrm{X}^T\mathrm{X})^{-1}\,\mathrm{X}^T\mathrm{y}$.

Mô hình hồi quy tuyến tính

Ví dụ: Xây dựng mô hình hồi quy tuyến tính đơn cho dữ liệu về số năm kinh nghiệm và thu nhập dưới đây:

Số năm KN	1.1	1.3	1.5	2.0	2.2
Thu nhập	39343	46205	37731	43525	39891

$$X = \begin{pmatrix} 1 & 1.1 \\ 1 & 1.3 \\ 1 & 1.5 \\ 1 & 2.0 \\ 1 & 2.2 \end{pmatrix}, \; y = \begin{pmatrix} 39343 \\ 46205 \\ 37731 \\ 43525 \\ 39891 \end{pmatrix} \Rightarrow \begin{pmatrix} 5 & 8.1 \\ 8.1 & 13.99 \end{pmatrix} B = \begin{pmatrix} 206695 \\ 334750.5 \end{pmatrix}$$

$$\Rightarrow B = \begin{pmatrix} 41517.05 \\ -109.91 \end{pmatrix}$$

Đường hồi quy tuyến tính có dạng: $y = 41517.05 - 109.91\,x$

Mô hình hồi quy tuyến tính

```python
import numpy as np
import matplotlib.pyplot as plt

# Dữ liệu
x=np.array([1.1,1.3,1.5,2.0,2.2])
y=np.array([39343,46205,37731,43525,39891])

# Nghiệm bình phương tối thiểu
A=np.vstack([np.ones(len(x)),x]).T
b0,b1=np.linalg.lstsq(A,y,rcond=None)[0]
print("y=%f+%f x"%(b0,b1))
```

$$y = 41517.050691 - 109.907834\,x$$

Thực hành 3

Bài tập 3.1

Hãy thay đổi chương trình trong ví dụ phân tích ma trận $X = W H^T$ với hàm mất mát hiệu chỉnh có dạng như sau:

$$\mathcal{L}(W, H) = \sum_{(u,i):r_{ui}=1} \left(x_{ui} - \sum_{j=1}^{K} w_{uj} h_{ij} \right)^2 + \lambda \left(\|W\|_F^2 + \|H\|_F^2 \right)$$

Thực hành 3

Bài tập 3.2

Tìm hàm hồi quy có dạng:

$$y = \beta_0 + \beta_1 \sin(x) + \beta_2 \cos(x)$$

phù hợp nhất với bộ dữ liệu dưới đây:

x	1	2	3	4	5	6	7	8	9	10
y	3.23	-3.00	-3.84	-2.08	-0.93	4.52	10.94	6.44	-5.74	-7.52

Thực hành

Bài tập 3.3

Tải dữ liệu về (số năm kinh nghiệm,thu nhập) từ file *salary_data.csv* vào mảng bằng lệnh sau:

```
import pandas as pd
data=pd.read_csv("salary_data.csv")
x=data["YearsExperience"].values
y=data["Salary"].values
```

Xây dựng hàm hồi quy tuyến tính phù hợp nhất với dữ liệu, biểu diễn thu nhập theo số năm kinh nghiệm và dự báo thu nhập trung bình nếu số năm kinh nghiệm $x_0 = 10$.