Homework 6

201900800302 赵淇涛

1. Graphical Model

1. The expression for $P(S_i)$ in terms of un-factorized joint distribution is as follows:

$$P(S_1)=\sum_G\sum_E\sum_H\sum_{D_1}\sum_{D_2}\sum_{D_3}\sum_{S_2}P(G,E,H,D_1,D_2,D_3,S_1,S_2).$$

2. The expression for $P(S_i)$ in terms of factorized joint distribution reads

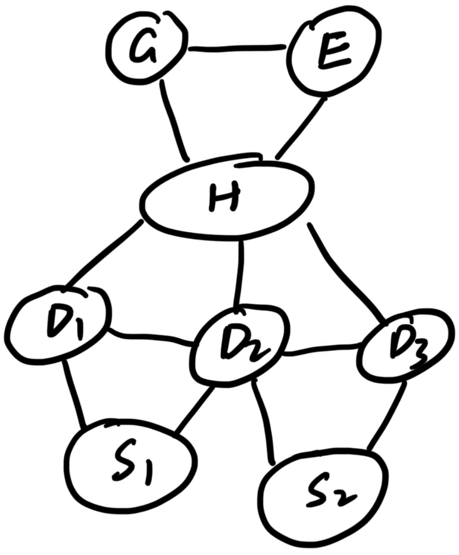
$$P(S_1)=\sum_G\sum_E\sum_H\sum_{D_1}\sum_{D_2}\sum_{D_3}\sum_{S_2}P(G)P(E)P(H|G,E)P(D_1|H)P(D_2|H)P(D_3|H)P(S_1|D_1,D_2)P(S_2|D_2,D_3).$$

The simplification procedure is shown below.

$$\begin{aligned}
P(S_1)&=\sum_E\sum_H\sum_{D_1}\sum_{D_2}\sum_{D_3}\sum_{S_2}P(E)P(D_1|H)P(D_2|H)P(D_3|H)P(S_1|D_1,D_2)P(S_2|D_2,D_3)\sum_G P(G)P(H|G,E)\\
&=\sum_H\sum_{D_1}\sum_{D_2}\sum_{D_3}\sum_{S_2}P(D_1|H)P(D_2|H)P(D_3|H)P(S_1|D_1,D_2)P(S_2|D_2,D_3)\sum_E m_G(H,E)P(E)\\
&=\sum_{D_1}\sum_{D_2}\sum_{D_3}\sum_{S_2}P(S_1|D_1,D_2)P(S_2|D_2,D_3)\sum_H m_E(H)P(D_1|H)P(D_2|H)P(D_3|H)\\
&=\sum_{D_2}\sum_{D_3}\sum_{S_2}P(S_2|D_2,D_3)\sum_{D_1}m_H(D_1,D_2,D_3)P(S_1|D_1,D_2)\\
&=\sum_{D_3}\sum_{S_2}\sum_{D_2}m_{D_1}(S_1,D_2,D_3)P(S_2|D_2,D_3)\\
&=\sum_{S_2}\sum_{D_3}m_{D_2}(S_1,S_2,D_3)\\
&=\sum_{S_2}m_{D_3}(S_1,S_2)\\
&=m_{S_2}(S_1)
\end{aligned}$$

3. The result in *2.* corresponds to the elimination order $GEHD_1D_2D_3S_2$.
4. The variable elimination algorithm is rewritten in this part.

   The moralized graph:



   The expression for $P(S_1)$ is rewritten as

$$P(S_1)=\sum_G\sum_E\sum_H\sum_{D_1}\sum_{D_2}\sum_{D_3}\sum_{S_2}\phi_1(G)\phi_2(E)\phi_3(H,G,E)\phi_4(D_1,H)\phi_5(D_2,H)\phi_6(D_3,H)\phi_7(S_1,D_1,D_2)\phi_8(S_2,D_2,D_3).$$

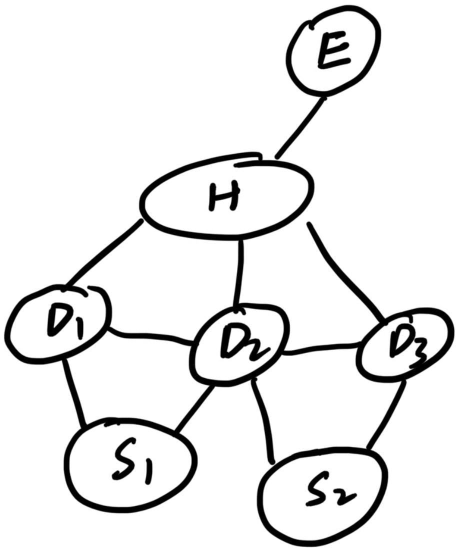
   Initial set of factors: $\{\phi_1(G),\phi_2(E),\phi_3(H,G,E),\phi_4(D_1,H),\phi_5(D_2,H),\phi_6(D_3,H),\phi_7(S_1,D_1,D_2),\phi_8(S_2,D_2,D_3)\}$

Step 1 (G):

$$\begin{aligned}P(S_1) &= \sum_E \sum_H \sum_{D_1} \sum_{D_2} \sum_{D_3} \sum_{S_2} \phi_2(E)\phi_4(D_1,H)\phi_5(D_2,H)\phi_6(D_3,H)\phi_7(S_1,D_1,D_2)\phi_8(S_2,D_2,D_3) \sum_G \phi_1(G)\phi_3(H,G,E) \\ &= \sum_E \sum_H \sum_{D_1} \sum_{D_2} \sum_{D_3} \sum_{S_2} \phi_2(E)\phi_4(D_1,H)\phi_5(D_2,H)\phi_6(D_3,H)\phi_7(S_1,D_1,D_2)\phi_8(S_2,D_2,D_3) \sum_G \psi_1(H,G,E) \\ &= \sum_E \sum_H \sum_{D_1} \sum_{D_2} \sum_{D_3} \sum_{S_2} \phi_2(E)\phi_4(D_1,H)\phi_5(D_2,H)\phi_6(D_3,H)\phi_7(S_1,D_1,D_2)\phi_8(S_2,D_2,D_3)\tau_1(H,E).\end{aligned}$$

Remaining factors: $\{\phi_2(E), \phi_4(D_1,H), \phi_5(D_2,H), \phi_6(D_3,H), \phi_7(S_1,D_1,D_2), \phi_8(S_2,D_2,D_3), \tau_1(H,E)\}$

New graph:

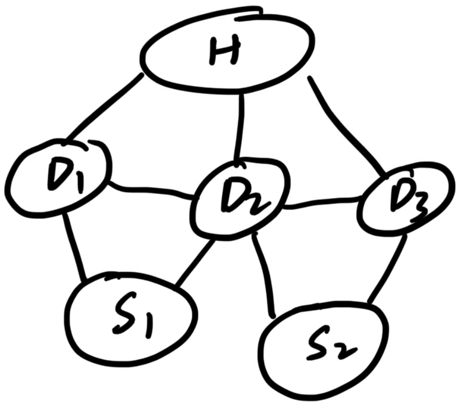


Step 2 (E):

$$\begin{aligned}P(S_1) &= \sum_H \sum_{D_1} \sum_{D_2} \sum_{D_3} \sum_{S_2} \phi_4(D_1,H)\phi_5(D_2,H)\phi_6(D_3,H)\phi_7(S_1,D_1,D_2)\phi_8(S_2,D_2,D_3) \sum_E \tau_1(H,E)\phi_2(E) \\ &= \sum_H \sum_{D_1} \sum_{D_2} \sum_{D_3} \sum_{S_2} \phi_4(D_1,H)\phi_5(D_2,H)\phi_6(D_3,H)\phi_7(S_1,D_1,D_2)\phi_8(S_2,D_2,D_3) \sum_E \psi_2(H,E) \\ &= \sum_H \sum_{D_1} \sum_{D_2} \sum_{D_3} \sum_{S_2} \phi_4(D_1,H)\phi_5(D_2,H)\phi_6(D_3,H)\phi_7(S_1,D_1,D_2)\phi_8(S_2,D_2,D_3)\tau_2(H).\end{aligned}$$

Remaining factors: $\{\phi_4(D_1,H), \phi_5(D_2,H), \phi_6(D_3,H), \phi_7(S_1,D_1,D_2), \phi_8(S_2,D_2,D_3), \tau_2(H)\}$

New graph:

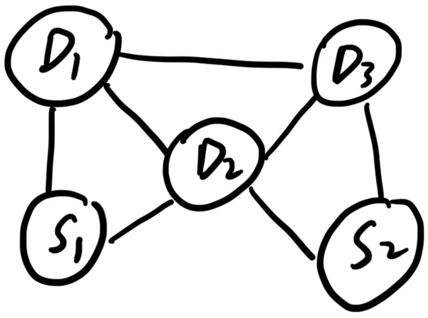


Step 3 (H):

$$\begin{aligned}P(S_1) &= \sum_{D_1}\sum_{D_2}\sum_{D_3}\sum_{S_2}\phi_7(S_1,D_1,D_2)\phi_8(S_2,D_2,D_3)\sum_{H}\tau_2(H)\phi_4(D_1,H)\phi_5(D_2,H)\phi_6(D_3,H)\\ &= \sum_{D_1}\sum_{D_2}\sum_{D_3}\sum_{S_2}\phi_7(S_1,D_1,D_2)\phi_8(S_2,D_2,D_3)\sum_{H}\psi_3(H,D_1,D_2,D_3)\\ &= \sum_{D_1}\sum_{D_2}\sum_{D_3}\sum_{S_2}\phi_7(S_1,D_1,D_2)\phi_8(S_2,D_2,D_3)\tau_3(D_1,D_2,D_3).\end{aligned}$$

Remaining factors: $\{\phi_7(S_1,D_1,D_2),\phi_8(S_2,D_2,D_3),\tau_3(D_1,D_2,D_3)\}$

New graph:

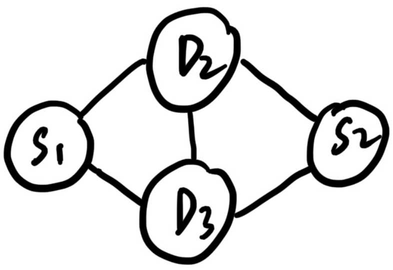


Step 4 (D_1):

$$\begin{aligned}P(S_1) &= \sum_{D_2}\sum_{D_3}\sum_{S_2}\phi_8(S_2,D_2,D_3)\sum_{D_1}\tau_3(D_1,D_2,D_3)\phi_7(S_1,D_1,D_2)\\ &= \sum_{D_2}\sum_{D_3}\sum_{S_2}\phi_8(S_2,D_2,D_3)\sum_{D_1}\psi_4(S_1,D_1,D_2,D_3)\\ &= \sum_{D_2}\sum_{D_3}\sum_{S_2}\phi_8(S_2,D_2,D_3)\tau_4(S_1,D_2,D_3).\end{aligned}$$

Remaining factors: $\{\phi_8(S_2,D_2,D_3),\tau_4(S_1,D_2,D_3)\}$

New graph:

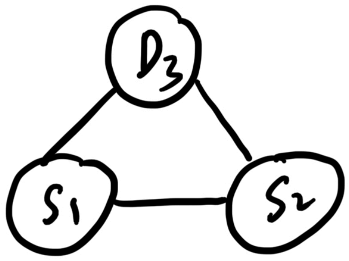


Step 5 (D_2):

$$\begin{aligned}P(S_1) &= \sum_{D_3}\sum_{S_2}\sum_{D_2}\phi_8(S_2,D_2,D_3)\tau_4(S_1,D_2,D_3)\\ &= \sum_{D_3}\sum_{S_2}\sum_{D_2}\psi_5(S_1,S_2,D_2,D_3)\\ &= \sum_{D_3}\sum_{S_2}\tau_5(S_1,S_2,D_3).\end{aligned}$$

Remaining factor: $\{\tau_5(S_1,S_2,D_3)\}$

New graph:

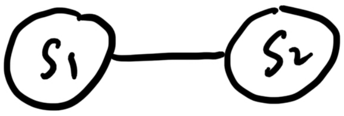


Step 6 (D_3):

$$\begin{aligned} P(S_1) &= \sum_{S_2}\sum_{D_3} \tau_5(S_1, S_2, D_3) \\ &= \sum_{S_2} \tau_6(S_1, S_2). \end{aligned}$$

Remaining factor: $\{\tau_6(S_1, S_2)\}$

New graph:



Step 7 (S_2):

$$\begin{aligned} P(S_1) &= \sum_{S_2} \tau_6(S_1, S_2) \\ &= \tau_7(S_1). \end{aligned}$$

Remaining factor: $\{\tau_7(S_1)\}$

Final graph:

5. The size of the largest intermediate factor $(\psi_4(S_1, D_1, D_2, D_3))$ is 4. Considering each variable can take m values, the factor has m^4 entries.
6. Given the largest size of the intermediate facter, the computational complexity of variable elimination is $O(nm^4)$.
7. Empirically, the order $\{S_2, D_3, G, E, H, D_1, D_2\}$ gives a lower computational complexity $(O(nm^3))$ because the largest size of the intermediate factor in this case is 3.

2. Recommender System

Overview

In many cases, we would like to recommend some items to users. A typical method is to do this on the basis of existing scores that users give to some of the items.

Here we would recommend movies to viewers. The algorithm of **Collaborative filtering** is introduced here. All users and movies are packed into a matrix, each element of which denotes the mark provided by the corresponding user to the movie. While we have some marks given by users, the matrix is not fully filled because it's not pracitical that users have watched all movies. Hence, we need to estimate the missing marks, by which we could do recommendation for users.

We would approximate the matrix by the product of two matrices, i.e., the user matrix and the movie matrix. The elements of two matrices can be explained as specific user behaviors related to their preferences or as movie attributes affecting viewers' choices. The gradient descent is adopted to minimize the differences between the critics matrix and the product of the user matrix and the movies matrix, specifically for the known elements and estimated elements are to be used to do recommendation.

Algorithm in detail

Consider a critics matrix **Z**. Its elements is given by

$$\mathbf{Z}_{ij} = r_{ij},$$

where r_{ij} is either the observed mark or zero as it is not given.

The user matrix **U** is of the size NxK and the movie matrix **V** is of the size KxM, where N denotes the number of users, M is the number of movies, K is a hyperparameter.

Let $\mathbf{A} = \{(i,j) : r_{ij}\ is\ observed\}$. The optimization objective **J** reads

$$\mathbf{J} = \frac{1}{2}\sum_{(i,j)\in A}(r_{ij} - \mathbf{U}_i\mathbf{V}_j)^2.$$

In order to implement gradient descent, calculate the gradient of **U** and **V** w.r.t. **J** as

$$\bigtriangledown_{\mathbf{U}}\mathbf{J} = -(\mathbf{Z} - \mathbf{U}\mathbf{V}) \cdot \mathbf{V}^T,$$
$$\bigtriangledown_{\mathbf{V}}\mathbf{J} = -\mathbf{U}^T \cdot (\mathbf{Z} - \mathbf{U}\mathbf{V}).$$

Note that an intermediate matrix $(\mathbf{Z} - \mathbf{U}\mathbf{V})$ is defined to do the dot production. Before the calculation of the gradients, its elements corresponding to zeros in matrix $\mathbf{Z}$ are set to zero. This is because we only want to take none-zero elements in **Z** into account while calculating gradients.

To make our model simple and more robust, we can also add normalization terms, e.g., L2 norms of matrices $\mathbf{U}$ and $\mathbf{V}$.

Code and process

0. Set up

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

1. Function *MF* to do matrix factorization

```
def MF(critics_map, epoch, k, lr, weight_decay):
    U, M = critics_map.shape
    user = np.random.normal(0, 1, (U, k))
    movie = np.random.normal(0, 1, (k, M))
    mask = critics_map == 0
    loss_history = []

    for i in range(epoch):
        temp = critics_map - user.dot(movie)
        temp[mask] = 0
        loss = 0.5 * np.linalg.norm(temp) ** 2
        loss += 0.5 * weight_decay * (np.linalg.norm(user)**2 + np.linalg.norm(user)**2)

        grad_u = -temp.dot(movie.T) + weight_decay * user
        grad_m = -user.T.dot(temp) + weight_decay * movie

        user -= lr * grad_u
        movie -= lr * grad_m

        if i % 500 == 0:
            loss_history.append(loss)
            print("Epoch: %d, loss: %.4f" % (i, loss))

            if i != 0:
                if abs(loss_history[-1] - loss_history[-2]) < 1e-6:
                    break

    return user.dot(movie)
```

2. Data pre-processing

```
users = ['Lisa Rose', 'Gene Seymour', 'Michael Phillips', 'Claudia Puig',
         'Mick LaSalle', 'Jack Matthews', 'Toby']

movies = ["Lady in the Water", "Snakes on a Plane", "Just My Luck",
          "Superman Returns", "You, Me and Dupree", "The Night Listener"]

critics = {'Lisa Rose': {'Lady in the Water': 2.5, 'Snakes on a Plane': 3.5,
'Just My Luck': 3.0, 'Superman Returns': 3.5, 'You, Me and Dupree': 2.5,
'The Night Listener': 3.0},
'Gene Seymour': {'Lady in the Water': 3.0, 'Snakes on a Plane': 3.5,
'Just My Luck': 1.5, 'Superman Returns': 5.0, 'You, Me and Dupree': 3.5,
'The Night Listener': 3.0},
'Michael Phillips': {'Lady in the Water': 2.5, 'Snakes on a Plane': 3.0,
'Superman Returns': 3.5, 'The Night Listener': 4.0},
'Claudia Puig': {'Snakes on a Plane': 3.5, 'Just My Luck': 3.0,
'Superman Returns': 4.0, 'You, Me and Dupree': 2.5, 'The Night Listener': 4.5},
'Mick LaSalle': {'Lady in the Water': 3.0, 'Snakes on a Plane': 4.0,
'Just My Luck': 2.0, 'Superman Returns': 3.0, 'You, Me and Dupree': 2.0,
'The Night Listener': 3.0},
'Jack Matthews': {'Lady in the Water': 3.0, 'Snakes on a Plane': 4.0,
'Superman Returns': 5.0, 'You, Me and Dupree': 3.5, 'The Night Listener': 3.0,},
'Toby': {'Snakes on a Plane': 4.5, 'Superman Returns': 4.0, 'You, Me and Dupree': 1.0,}}

# Pre-processing
critics_map = np.zeros((7, 6))

for i in range(7):
    for j in range(6):
        critics_map[i, j] = critics[users[i]].get(movies[j], 0)
```

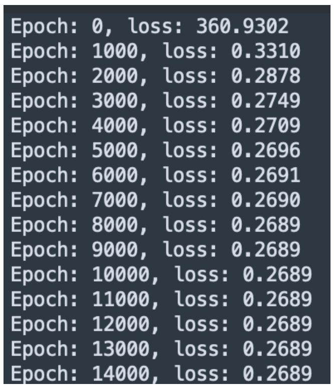
3. Main part

```
mask = critics_map != 0

epoch = 200000
k = 10
learning_rate = 0.005
weight_decay = 0.01
predicted_critics_map = MF(critics_map, epoch, k, learning_rate, weight_decay)
print(predicted_critics_map)
predicted_critics_map[mask] = 0
print("'%s' would be recommended for Toby.\nIt got %0.2f!" % (movies[np.argmax(predicted_critics_map[-1])],
np.max(predicted_critics_map[-1])))
```

Result and discussion

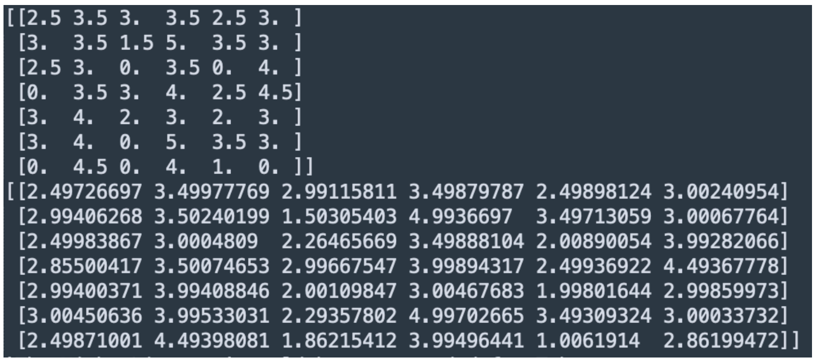
1. Loss history



```
Epoch: 0, loss: 360.9302
Epoch: 1000, loss: 0.3310
Epoch: 2000, loss: 0.2878
Epoch: 3000, loss: 0.2749
Epoch: 4000, loss: 0.2709
Epoch: 5000, loss: 0.2696
Epoch: 6000, loss: 0.2691
Epoch: 7000, loss: 0.2690
Epoch: 8000, loss: 0.2689
Epoch: 9000, loss: 0.2689
Epoch: 10000, loss: 0.2689
Epoch: 11000, loss: 0.2689
Epoch: 12000, loss: 0.2689
Epoch: 13000, loss: 0.2689
Epoch: 14000, loss: 0.2689
```

The loss drops drastically in first many steps and takes a while to converge.

2. The estimated matrix



```
[[2.5 3.5 3.  3.5 2.5 3. ]
 [3.  3.5 1.5 5.  3.5 3. ]
 [2.5 3.  0.  3.5 0.  4. ]
 [0.  3.5 3.  4.  2.5 4.5]
 [3.  4.  2.  3.  2.  3. ]
 [3.  4.  0.  5.  3.5 3. ]
 [0.  4.5 0.  4.  1.  0. ]]
[[2.49726697 3.49977769 2.99115811 3.49879787 2.49898124 3.00240954]
 [2.99406268 3.50240199 1.50305403 4.9936697  3.49713059 3.00067764]
 [2.49983867 3.0004809  2.26465669 3.49888104 2.00890054 3.99282066]
 [2.85500417 3.50074653 2.99667547 3.99894317 2.49936922 4.49367778]
 [2.99400371 3.99408846 2.00109847 3.00467683 1.99801644 2.99859973]
 [3.00450636 3.99533031 2.29357802 4.99702665 3.49309324 3.00033732]
 [2.49871001 4.49398081 1.86215412 3.99496441 1.0061914  2.86199472]]
```

The original critics matrix $\mathbf{Z}$ and the estimated one are shown above. Note that all missing marks are filled by predicted values and existing values in $\mathbf{Z}$ are well approximated. The last row corresponds to the target user Toby. Except for those movies Toby has watched, the last item achieves the highest score, which denotes the estimated mark for the movie 'The Night Listener'. Hence, this movie would be recommended for Toby.