Lab Report
Data & Knowledge - Factorization-Based Data Modeling
Practical Work 3

ZHANG Xin

January 2020

1 Develop a (non-negative) coupled factorization model for decomposing all these observed matrices/tensors simultaneously.

Suppose that we have $X_1 \in \mathbb{R}^{m\times n\times r}$, $X_2 \in \mathbb{R}^{m\times n}$, $X_3 \in \mathbb{R}^{n\times p}$, $X_4 \in \mathbb{R}^{m\times m}$, $X_5 \in \mathbb{R}^{r\times r}$.
So we adapt a PARAFAC-style approach, define a rank k, for the decomposition of X_1, we can have three low-dimensional representations:
User $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k] \in \mathbb{R}^{m\times k}$, Locations $L = [\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_k] \in \mathbb{R}^{n\times k}$ and Activities $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_k] \in \mathbb{R}^{r\times k}$, User 2 $U_2 \in \mathbb{R}^{k\times m}$, Activities 2 $A_2 \in \mathbb{R}^{k\times r}$ and Locations 2 $L_2 \in \mathbb{R}^{k\times n}$
And for X_3, in addition to L, we define the Feature $F = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_k] \in \mathbb{R}^{p\times k}$.
So we have:

$$\begin{aligned}
\hat{X}_1 &= [\![U, L, A]\!] = \Sigma_{i=1}^{k} \mathbf{u}_i \otimes \mathbf{l}_i \otimes \mathbf{a}_i \\
\hat{X}_2 &= UL_2 \\
\hat{X}_3 &= LF^T \\
\hat{X}_4 &= UU_2 \\
\hat{X}_5 &= AA_2
\end{aligned}$$

where $\otimes$ denotes the outer product.

2 Write down the cost function by using the β-divergence

The cost function

$$\begin{aligned}
\mathcal{L}(U, L, A, F) &= D_{\beta_1}(X_1 \| [\![U, L, A]\!]) + \lambda_2 D_{\beta_2}(X_2 \| UL_2) + \lambda_3 D_{\beta_3}(X_3 \| LF^T) + \lambda_4 D_{\beta_4}(X_4 \| UU_2) + \lambda_5 D_{\beta_5}(X_5 \| AA_2) \\
&= \Sigma_{m=1}^{M} \Sigma_{n=1}^{N} \Sigma_{r=1}^{R} d([X_1]_{mnr} \| [\![U, L, A]\!]_{mnr}) + \lambda_2 \Sigma_{m=1}^{M} \Sigma_{n=1}^{N} d([X_2]_{mn} \| [UL_2]_{mn}) + \lambda_3 \Sigma_{n=1}^{N} \Sigma_{p=1}^{P} d([X_3]_{np} \| [LF^T]_{np}) \\
&+ \lambda_4 \Sigma_{m=1}^{M} \Sigma_{m=1}^{M} d([X_4]_{mm} \| [UU_2]_{mm}) + \lambda_5 \Sigma_{r=1}^{R} \Sigma_{r=1}^{R} d([X_5]_{rr} \| [AA_2]_{rr})
\end{aligned}$$

3 Explain why the model makes sense.

The model makes sense because that when $X_{1:5} = \hat{X}_{1:5}$, the β-divergence $d_\beta(x \| \hat{x}) = \frac{x^\beta}{\beta(\beta-1)} - \frac{x\hat{x}^{\beta-1}}{\beta-1} + \frac{\hat{x}^\beta}{\beta} = 0$. So if we minimize the cost function defined in question 2 to 0, it will also minimize the difference between the model and the real dataset. So we have the optimization problem

$$(U^*, L^*, A^*, F^*) = \underset{U,L,A,F \geq 0}{\arg\min}\ \mathcal{L}(U, L, A, F)$$

4 Develop the multiplicative update rules algorithm for the model

Here we try to apply gradient descend for the model:
We have:

$$\nabla_U = (\hat{X_1}^{\cdot\beta_1-1} - X_1 \cdot \hat{X_1}^{\cdot\beta_1-2})^{(1)}(A * L) + \lambda_2(\hat{X_2}^{\cdot\beta_2-1} - X_2 \cdot \hat{X_2}^{\cdot\beta_2-2})L_2^T + \lambda_4(\hat{X_4}^{\cdot\beta_4-1} - X_4 \cdot \hat{X_4}^{\cdot\beta_4-2})U_2^T$$

$$\nabla_L = (\hat{X_1}^{\cdot\beta_1-1} - X_1 \cdot \hat{X_1}^{\cdot\beta_1-2})^{(2)}(A * U) + \lambda_3(\hat{X_3}^{\cdot\beta_3-1} - X_3 \cdot \hat{X_3}^{\cdot\beta_3-2})F$$

$$\nabla_A = (\hat{X_1}^{\cdot\beta_1-1} - X_1 \cdot \hat{X_1}^{\cdot\beta_1-2})^{(3)}(L * U) + \lambda_5(\hat{X_5}^{\cdot\beta_5-1} - X_5 \cdot \hat{X_5}^{\cdot\beta_5-2})A_2^T$$

$$\nabla_F = \lambda_3(\hat{X_3}^{\cdot\beta_3-1} - X_3 \cdot \hat{X_3}^{\cdot\beta_3-2})^T L$$

$$\nabla_{U_2} = \lambda_4 U^T(\hat{X_4}^{\cdot\beta_4-1} - X_4 \cdot \hat{X_4}^{\cdot\beta_4-2})$$

$$\nabla_{L_2} = \lambda_2 U^T(\hat{X_2}^{\cdot\beta_2-1} - X_2 \cdot \hat{X_2}^{\cdot\beta_2-2})$$

$$\nabla_{A_2} = \lambda_5 A^T(\hat{X_5}^{\cdot\beta_5-1} - X_5 \cdot \hat{X_5}^{\cdot\beta_5-2})$$

where $\cdot^{(i)}$ means the mode-i of a tensor, $*$ means the Kronecker product, and $\cdot$ means the outer product.
So the update rule is:

$$U_{t+1} = U_t - \gamma\nabla_U$$

$$L_{t+1} = L_t - \gamma\nabla_L$$

$$A_{t+1} = A_t - \gamma\nabla_A$$

$$F_{t+1} = F_t - \gamma\nabla_F$$

$$U2_{t+1} = U2_t - \gamma\nabla_{U2}$$

$$L2_{t+1} = L2_t - \gamma\nabla_{L2}$$

$$A2_{t+1} = A2_t - \gamma\nabla_{A2}$$

By choosing proper step-size γ for each update rule, we can get the multiplicative update rules:

$$U_{t+1} = U_t \cdot \frac{(X_1 \cdot \hat{X_1}^{\cdot\beta_1-2})^{(1)}(A * L) + \lambda_2 X_2 \cdot \hat{X_2}^{\cdot\beta_2-2} L_2^T + \lambda_4 X_4 \cdot \hat{X_4}^{\cdot\beta_4-2} U_2^T}{(\hat{X_1}^{\cdot\beta_1-1})^{(1)}(A * L) + \lambda_2 \hat{X_2}^{\cdot\beta_2-1} L_2^T + \lambda_4 \hat{X_4}^{\cdot\beta_4-1} U_2^T}$$

$$L_{t+1} = L_t \cdot \frac{(X_1 \cdot \hat{X_1}^{\cdot\beta_1-2})^{(2)}(A * U) + \lambda_3 X_3 \cdot \hat{X_3}^{\cdot\beta_3-2} F}{(\hat{X_1}^{\cdot\beta_1-1})^{(2)}(A * U) + \lambda_3 \hat{X_3}^{\cdot\beta_3-1} F}$$

$$U_{t+1} = U_t \cdot \frac{(X_1 \cdot \hat{X_1}^{\cdot\beta_1-2})^{(3)}(L * U) + \lambda_5 X_5 \cdot \hat{X_5}^{\cdot\beta_5-2} A_2^T}{(\hat{X_1}^{\cdot\beta_1-1})^{(3)}(L * U) + \lambda_3 \hat{X_3}^{\cdot\beta_3-1} A_2^T}$$

$$F_{t+1} = L_t \cdot \lambda_3 \frac{X_3^T L}{\hat{X_3}^T L}$$

$$U2_{t+1} = U2_t \cdot \lambda_4 \frac{U^T X_4}{U^T \hat{X}_4}$$

$$L2_{t+1} = L2_t \cdot \lambda_2 \frac{U^T X_2}{U^T \hat{X}_2}$$

$$A2_{t+1} = A2_t \cdot \lambda_5 \frac{A^T X_5}{A^T \hat{X}_5}$$

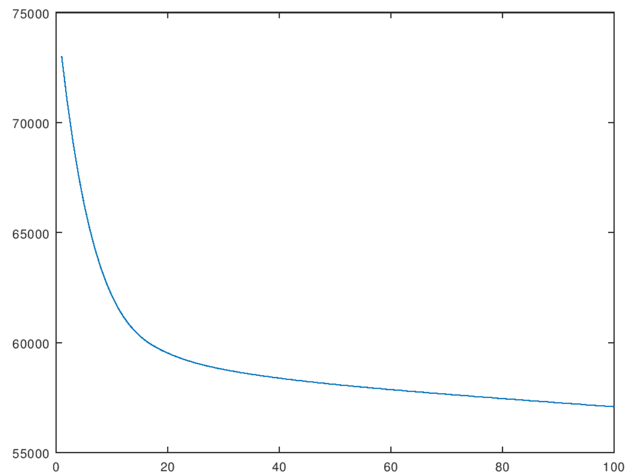
5 Implement your algorithm in Octave. Monitor the overall cost function. What are the effect of choosing different β for each tensor? When the algorithm converges, check whether the individual model predictions $\hat{X}_{1:5}$ are close to the original tensors or not.

The implementation of the additive update rule is as follows:

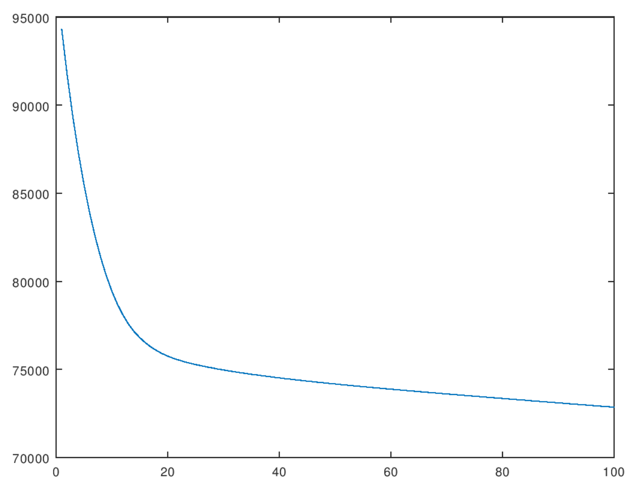
```
clear
close all
clc

load('ucla f_data.mat');

beta = [1.5, 0.5, 0.5, 0.5, 0.5]; %beta for every tensor
w = [1, 0.1, 0.1, 0.1, 0.1]; %the weight of each tensor for the loss function

alpha = 0.00001;
k = 3;
nIter = 100;

U = rand(size(UserLocAct, 1),k);
L = rand(size(UserLocAct, 2),k);
A = rand(size(UserLocAct, 3),k);
F = rand(size(LocFea, 2), k);
U2 = rand(k, size(UserLocAct, 1));
A2 = rand(k, size(UserLocAct, 3));
L2 = rand(k, size(UserLocAct, 2));

ULAhat = zeros(size(UserLocAct));
for j = 1:size(UserLocAct, 3),  ULAhat(:,:,j) = U * diag(A(j,:)) * L'; end

loss1 = sum(sum(sum(UserLocAct.^(beta(1))./(beta(1).*(beta(1)-1)) - UserLocAct.*(ULAhat.^(beta(1)-1))./(beta(1)-1)+ULAhat.^(beta(1))./beta(1))));
loss2 = sum(sum(UserLoc.^(beta(2))./(beta(2).*(beta(2)-1))-UserLoc.*((U*L2).^(beta(2)-1))./(beta(2)-1)+((U*L2).^(beta(2)))./beta(2)));
loss3 = sum(sum(LocFea.^(beta(3))./(beta(3).*(beta(3)-1))-LocFea.*((L*F').^(beta(3)-1))./(beta(3)-1)+((L*F').^(beta(3)))./beta(3)));
loss4 = sum(sum(UserUser.^(beta(4))./(beta(4).*(beta(4)-1))-UserUser.*((U*U2).^(beta(4)-1))./(beta(4)-1)+((U*U2).^(beta(4)))./beta(4)));
loss5 = sum(sum(ActAct.^(beta(5))./(beta(5).*(beta(5)-1))-ActAct.*((A*A2).^(beta(5)-1))./(beta(5)-1)+((A*A2).^(beta(5)))./beta(5)));
Loss = abs(loss1+w(2)*loss2+w(3)*loss3+w(4)*loss4+w(5)*loss5);
oldLoss = Loss;
obj = zeros(1,nIter);

for it = 1:nIter

  dU = (tenmat((ULAhat.^(beta(1)-1)-UserLocAct.*(ULAhat.^(beta(1)-2))),1)*khatrirao(A,L)).data+w(2)*((U*L2).^(beta(2)-1)-UserLoc.*((U*L2).^(beta(2)-2)))*L2'+w(4)*((U*U2).^(beta(4)-1)-UserUser.*((U*U2).^(beta(4)-2)))*U2';
  dL = (tenmat((ULAhat.^(beta(1)-1)-UserLocAct.*(ULAhat.^(beta(1)-2))),2)*khatrirao(A,U)).data+w(3)*((L*F').^(beta(3)-1)-LocFea.*((L*F').^(beta(3)-2)))*F;
  dA = (tenmat((ULAhat.^(beta(1)-1)-UserLocAct.*(ULAhat.^(beta(1)-2))),3)*khatrirao(L,U)).data+w(5)*((A*A2).^(beta(5)-1)-ActAct.*((A*A2).^(beta(5)-2)))*A2';
  dF = w(3)*((L*F').^(beta(3)-1)-LocFea.*((L*F').^(beta(3)-2)))'*L;
  dU2 = w(4)*U'*((U*U2).^(beta(4)-1)-UserUser.*((U*U2).^(beta(4)-2)));
  dL2 = w(2)*U'*((U*L2).^(beta(2)-1)-UserLoc.*((U*L2).^(beta(2)-2)));
  dA2 = w(5)*A'*((A*A2).^(beta(5)-1)-ActAct.*((A*A2).^(beta(5)-2)));

    U = max(U - alpha*dU, 0);
    L = max(L - alpha*dL, 0);
    A = max(A - alpha*dA, 0);
    F = max(F - alpha*dF, 0);
    U2 = max(U2 - alpha*dU2, 0);
    L2 = max(L2 - alpha*dL2, 0);
    A2 = max(A2 - alpha*dA2, 0);

    for j = 1:size(UserLocAct, 3),  ULAhat(:,:,j) = U * diag(A(j,:)) * L'; end

  loss1 = sum(sum(sum(UserLocAct.^(beta(1))./(beta(1).*(beta(1)-1)) - UserLocAct.*(ULAhat.^(beta(1)-1))./(beta(1)-1)+ULAhat.^(beta(1))./beta(1))));
  loss2 = sum(sum(UserLoc.^(beta(2))./(beta(2).*(beta(2)-1))-UserLoc.*((U*L2).^(beta(2)-1))./(beta(2)-1)+((U*L2).^(beta(2)))./beta(2)));
```

```
  loss3 = sum(sum(LocFea.^(beta(3))./(beta(3).*(beta(3)-1))-LocFea.*((L*F').^(beta(3)-1))./(beta(3)-1)+((L*F').^(beta(3)))./beta(3)));
  loss4 = sum(sum(UserUser.^(beta(4))./(beta(4).*(beta(4)-1))-UserUser.*((U*U2).^(beta(4)-1))./(beta(4)-1)+((U*U2).^(beta(4)))./beta(4)));
  loss5 = sum(sum(ActAct.^(beta(5))./(beta(5).*(beta(5)-1))-ActAct.*((A*A2).^(beta(5)-1))./(beta(5)-1)+((A*A2).^(beta(5)))./beta(5)));
  Loss = abs(loss1+w(2)*loss2+w(3)*loss3+w(4)*loss4+w(5)*loss5);
  obj(it) = Loss;

end

figure,
plot(obj);
```

With the shown β values and 100 iterations, we can get a graph of the loss as follows:



If we change the β to `beta = [1.5, 1.5, 1.5, 0.3, 0.3];`, we get



We can see that the value of β doesn't change the convergence, but with bigger β we have bigger convergence value.
For the predictions, this method does not provide a close prediction, as the original tensors are really sparse with a lot of 0 as values, it is better to adapt a multiplicative approach.
The implementation of the multiplicative update rules is as follows, but it doesn't work for now because NaN values will appear in the multiplication process.

```
clear
close all
clc

load('uclaf_data.mat');
```

```
beta = [1.5, 3, 3, 3, 3]; %beta for every tensor
w = [1, 0.1, 0.1, 0.1, 0.1]; %the weight of each tensor for the loss function

%alpha = 0.00001;
k = 3;
nIter = 2;

U = rand(size(UserLocAct, 1),k);
L = rand(size(UserLocAct, 2),k);
A = rand(size(UserLocAct, 3),k);
F = rand(size(LocFea, 2), k);
U2 = rand(k, size(UserLocAct, 1));
A2 = rand(k, size(UserLocAct, 3));
L2 = rand(k, size(UserLocAct, 2));

ULAhat = zeros(size(UserLocAct));
for j = 1:size(UserLocAct, 3),  ULAhat(:,:,j) = U * diag(A(j,:)) * L'; end

loss1 = sum(sum(sum(UserLocAct.^(beta(1))./(beta(1).*(beta(1)-1)) - UserLocAct.*(ULAhat.^(beta(1)-1))./(beta(1)-1)+ULAhat.^(beta(1))./beta(1))));
loss2 = sum(sum(UserLoc.^(beta(2))./(beta(2).*(beta(2)-1))-UserLoc.*((U*L2).^(beta(2)-1))./(beta(2)-1)+((U*L2).^(beta(2)))./beta(2)));
loss3 = sum(sum(LocFea.^(beta(3))./(beta(3).*(beta(3)-1))-LocFea.*((L*F').^(beta(3)-1))./(beta(3)-1)+((L*F').^(beta(3)))./beta(3)));
loss4 = sum(sum(UserUser.^(beta(4))./(beta(4).*(beta(4)-1))-UserUser.*((U*U2).^(beta(4)-1))./(beta(4)-1)+((U*U2).^(beta(4)))./beta(4)));
loss5 = sum(sum(ActAct.^(beta(5))./(beta(5).*(beta(5)-1))-ActAct.*((A*A2).^(beta(5)-1))./(beta(5)-1)+((A*A2).^(beta(5)))./beta(5)));
Loss = abs(loss1+w(2)*loss2+w(3)*loss3+w(4)*loss4+w(5)*loss5);
oldLoss = Loss;
obj = zeros(1,nIter);

for it = 1:nIter

    U = U.*(((tenmat(UserLocAct.*(ULAhat.^(beta(1)-2)),1)*khatrirao(A,L)).data + w(2)*UserLoc.*((U*L2).^(beta(2)-2))*L2'+w(4)*UserUser.*((U*U2).^(beta(4)-2))*U2')./((tenmat(ULAhat.^(beta(1)-1),1)*khatrirao(A,L)).data+w(2)*(U*L2).^(beta(2)-1)*L2'+w(4)*(U*U2).^(beta(4)-1)*U2'));
    L = L.*(((tenmat(UserLocAct.*(ULAhat.^(beta(1)-2)),2)*khatrirao(A,U)).data + w(3)*LocFea.*((L*F').^(beta(3)-2))*F)./((tenmat(ULAhat.^(beta(1)-1),2)*khatrirao(A,U)).data+w(3)*(L*F').^(beta(3)-1)*F));
    A = A.*(((tenmat(UserLocAct.*(ULAhat.^(beta(1)-2)),3)*khatrirao(L,U)).data + w(5)*ActAct.*((A*A2).^(beta(5)-2))*A2')./((tenmat(ULAhat.^(beta(1)-1),3)*khatrirao(L,U)).data+w(5)*(A*A2).^(beta(5)-1)*A2'));
    F = F.*(w(3)*(LocFea'*L)./((L*F')'*L));
    U2 = U2.*(w(4)*(U'*UserUser)./(U'*U*U2));
    L2 = L2.*(w(2)*(U'*UserLoc)./(U'*U*L2));
    A2 = A2.*(w(5)*(A'*ActAct)./(A'*A*A2));

    for j = 1:size(UserLocAct, 3),  ULAhat(:,:,j) = U * diag(A(j,:)) * L'; end

  loss1 = sum(sum(sum(UserLocAct.^(beta(1))./(beta(1).*(beta(1)-1)) - UserLocAct.*(ULAhat.^(beta(1)-1))./(beta(1)-1)+ULAhat.^(beta(1))./beta(1))));
  loss2 = sum(sum(UserLoc.^(beta(2))./(beta(2).*(beta(2)-1))-UserLoc.*((U*L2).^(beta(2)-1))./(beta(2)-1)+((U*L2).^(beta(2)))./beta(2)));
  loss3 = sum(sum(LocFea.^(beta(3))./(beta(3).*(beta(3)-1))-LocFea.*((L*F').^(beta(3)-1))./(beta(3)-1)+((L*F').^(beta(3)))./beta(3)));
  loss4 = sum(sum(UserUser.^(beta(4))./(beta(4).*(beta(4)-1))-UserUser.*((U*U2).^(beta(4)-1))./(beta(4)-1)+((U*U2).^(beta(4)))./beta(4)));
  loss5 = sum(sum(ActAct.^(beta(5))./(beta(5).*(beta(5)-1))-ActAct.*((A*A2).^(beta(5)-1))./(beta(5)-1)+((A*A2).^(beta(5)))./beta(5)));
  Loss = loss1+w(2)*loss2+w(3)*loss3+w(4)*loss4+w(5)*loss5;
  obj(it) = Loss;

end

figure,
plot(obj);
```