

تمرین شماره ۵

درس تجزیه های تانسوری

امید شرفی

400201518

دکتر سپیده حاجی پور

June 23, 2022

سوال ۱

تابع HOOI را برای تانسور به ابعاد دلخواه به صورت زیر پیاده‌سازی کردیم.

```
function [G, U] = HOOI(T, rank, itr_n)

    % HOSVD for initial value
    U = cell(1, length(rank));
    for i = 1:length(rank)
        [U_temp, ~, ~] = svd(tenmat(T, i).data);
        U{i} = U_temp(:, 1:rank(i));
    end

    % Calculate U matrices using HOSVD and iteration
    for itr = 1:itr_n
        for k = 1:length(rank)
            temp_cell= cell(1,length(rank)-1);
            cnt = 1;
            for i = [1:k-1, k+1:length(rank)]
                temp_cell{cnt} = U{i}';
                cnt = cnt + 1;
            end
            [U_temp, ~, ~] = svd(ttm(T, temp_cell, [1:k-1, k+1:length(rank)]),k).data;
            U{k} = U_temp(:, 1:rank(k));
        end
    end

    % Calculate Core Matrix
    temp_cell = cell(1,length(rank));
    for i = 1:length(rank)
        temp_cell{i} = U{i}';
    end
    G = ttm(T, temp_cell, 1:length(rank));
end
```

برای تست صحت روش، یک تانسور با رتبه پایین‌تر از ابعاد هر بعد درست کرده و نتیجه‌ی تجزیه با رتبه‌های مربوطه به شرح زیر بوده که خطا قابل قبول و گویای صحت روش می‌باشد.

```
8      %% 1
9      % Test HOOI with small clear tensor
10 -     [U1, ~] = gsog(rand(7,5));
11 -     [U2, ~] = gsog(rand(8,7));
12 -     [U3, ~] = gsog(rand(10,4));
13
14 -     G = rand(5, 7, 4);
15 -     T = ttm(tensor(G), {U1, U2, U3}, 1:3);
16 -     rank = [5, 7, 4];
17 -     Itr = 1000;
18 -     [G, U] = HOOI(T, rank, Itr);
19
20 -     T_pred = ttm(G, U, 1:3);
21 -     error = sqrt(sum(double(T_pred-T).^2, 'all'));
22 -     fprintf('Frobenius norm error = %d, Iteration = %d\n', error, Itr);

Command Window
Frobenius norm error = 6.332979e-15, Iteration = 0
Frobenius norm error = 4.044193e-15, Iteration = 1
Frobenius norm error = 5.554387e-15, Iteration = 10
Frobenius norm error = 8.434813e-15, Iteration = 100
Frobenius norm error = 5.419004e-15. Iteration = 1000
```

```

8 %% 1
9 % Test HOOI with small clear tensor
10 - [U1, ~] = gsog(rand(7,5));
11 - [U2, ~] = gsog(rand(8,7));
12 - [U3, ~] = gsog(rand(10,4));
13
14 - G = rand(5, 7, 4);
15 - T = ttm(tensor(G), {U1, U2, U3}, 1:3);
16 - rank = [3, 5, 3];
17 - Itr = [0 1 10 100 1000];
18
19 - for itr = Itr
20 -
21 -     [G, U] = HOOI(T, rank, itr);
22 -
23 -     T_pred = ttm(G, U, 1:3);
24 -     error = sqrt(sum(double(T_pred-T).^2, 'all'));
25 -     fprintf('Frobenius norm error = %d, Iteration = %d\n', error, itr);
26 -
27 - end

```

Command Window

```

Frobenius norm error = 1.012591e+00, Iteration = 0
Frobenius norm error = 9.977676e-01, Iteration = 1
Frobenius norm error = 9.969472e-01, Iteration = 10
Frobenius norm error = 9.969472e-01, Iteration = 100
Frobenius norm error = 9.969472e-01, Iteration = 1000

```

مقادیر خطای رنک کمتر از مقادیر واقعی رنک

```

30 %% Test HOOI with Big clear tensor
31
32 - [U1, ~] = gsog(rand(30,20));
33 - [U2, ~] = gsog(rand(25,7));
34 - [U3, ~] = gsog(rand(40,24));
35
36 - G = rand(20, 7, 24);
37 - T_org = ttm(tensor(G), {U1, U2, U3}, 1:3);
38 - T = T_org + tensor(0.3*randn(30, 25, 40));
39 - rank = [20, 7, 24];
40 - Itr = 1:5;
41
42 - for itr = Itr
43 -
44 -     [G, U] = HOOI(T, rank, itr);
45 -
46 -     T_pred = ttm(G, U, 1:3);
47 -     error = sqrt(sum(double(T_pred-T).^2, 'all'));
48 -     fprintf('Frobenius norm error = %d, Iteration = %d\n', error, itr);
49 -
50 - end

```

Command Window

```

Frobenius norm error = 4.827567e+01, Iteration = 1
Frobenius norm error = 4.827483e+01, Iteration = 2
Frobenius norm error = 4.827482e+01, Iteration = 3
Frobenius norm error = 4.827482e+01, Iteration = 4
Frobenius norm error = 4.827482e+01, Iteration = 5

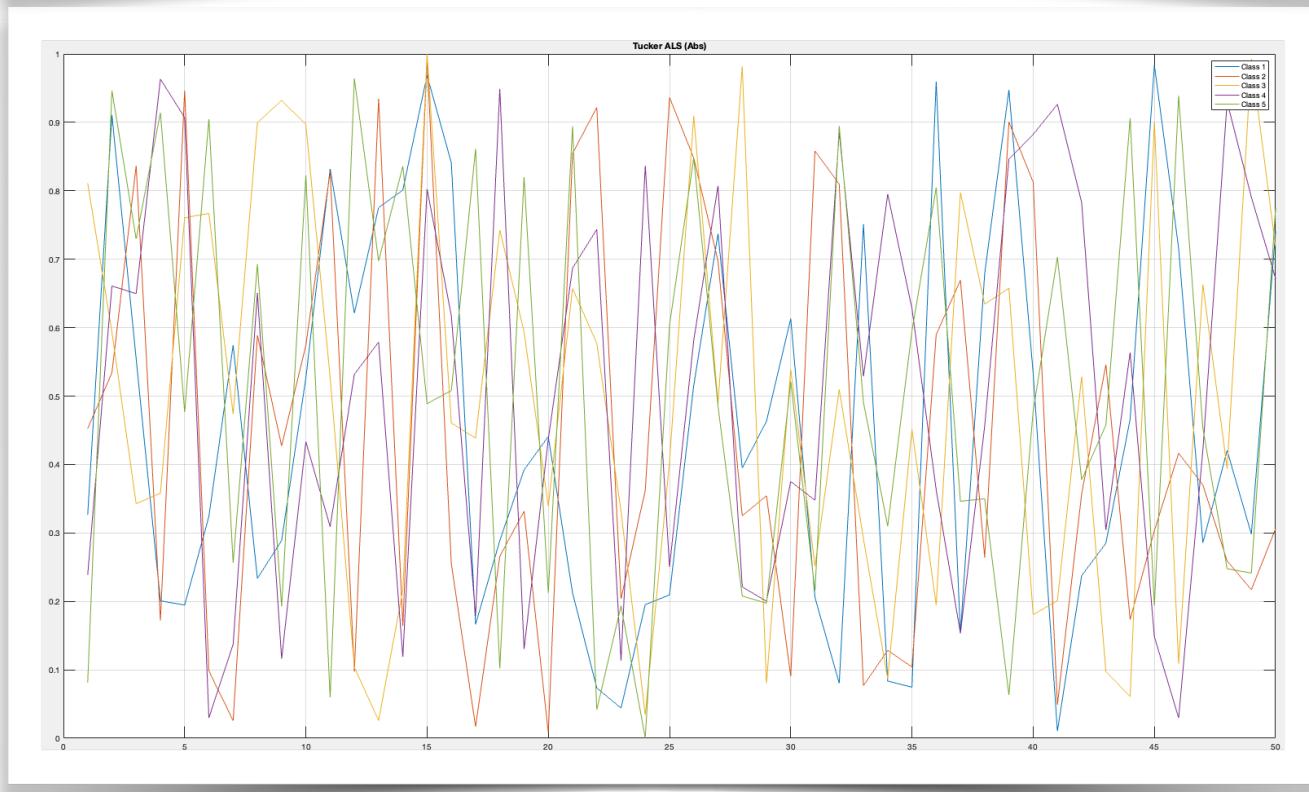
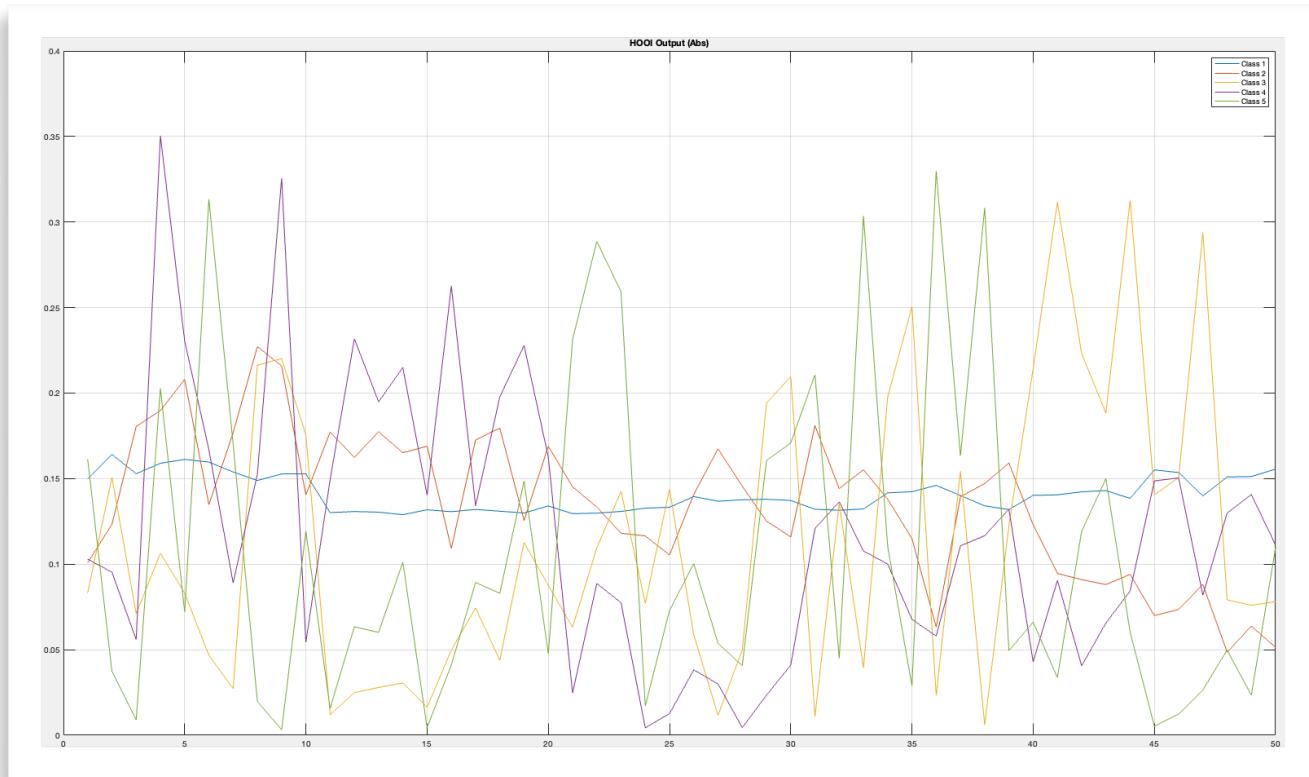
```

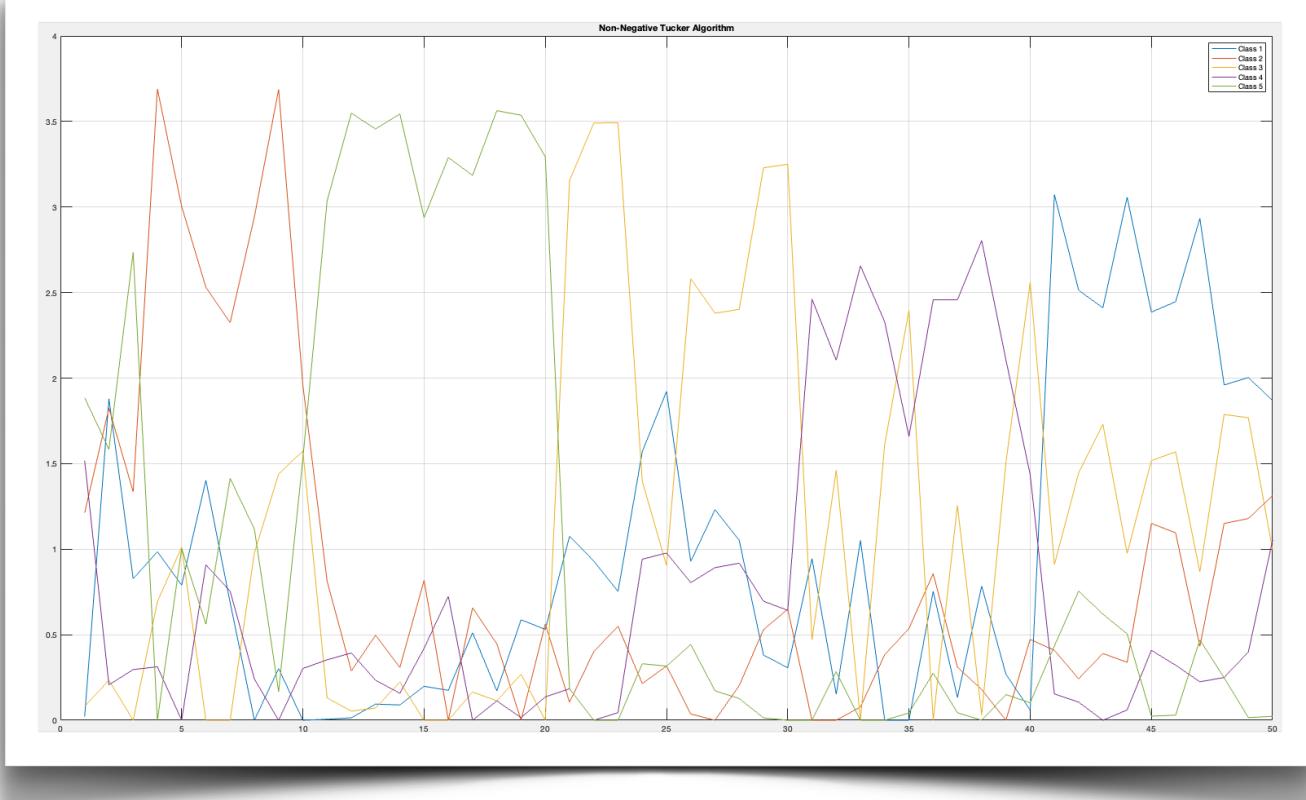
مقادیر خطای افزوده به تنسور بزرگ

همانطور که مشاهده می‌شود در تمام سناریوهای بالا الگوریتم HOOI پیاده‌سازی شده موفق

عمل می‌کند و میزان خطای بازسازی بسته به تعیین رنک، ابعاد تنسور، خطای اضافه شده به تنسور، و تعداد تکرارهایی که الگوریتم را اجرا می‌کنیم مشخص می‌گردد.

سؤال ۲





برای حل این مساله ابتدا باید تجزیه Tucker انجام شود و سپس عامل سوم که با مرتبه ۵ (معادل ۵ نفر(دسته)) در نظر گرفته می‌شود را بررسی و ترسیم کنیم.

در این مساله ما از دو تجزیه Non-Negative Tucker و ALS استفاده کردیم. برای انجام تجزیه‌ی Tucker از دو روش HOOI که خودمان پیاده سازی کردیم و روش ALS مطلب استفاده می‌کنیم. همچنین رنک عاملها در تمام روش‌های پیاده سازی شده یکسان و به ترتیب برابر با ۵، ۵۰، و ۵ در نظر گرفته شده است تا بتوانیم نتایج را با یکدیگر مقایسه کنیم. با این مقادیر روش HOOI به مراتب بهتر از ALS عمل کرده و با این که در تعدادی از تصاویر مخصوصاً مربوط به فرد سوم با خطأ همراه است، روند ۵ دسته‌ی ۱۰ تصویره در نتیجه مشاهده می‌شود. روش Non-Negative Tucker نتایج بسیار بهتری از دو روش دیگر به ما می‌دهد.

همانطور که مشاهده می‌شود روند ۱۰ داده اول برای یک دسته (فرد)، ۱۰ داده دوم برای دسته‌ی دیگر و به همین ترتیب برقرار است و صرفا در چند داده فرد سوم و چهارم دسته‌بندی درست انجام نشده است. علت این عملکرد بهتر را می‌توان به فرض مثبت بودن عامل‌ها نسبت داد که طبیعتاً برای مقادیر احتمالی که عامل سوم دارد فرض درستی می‌باشد و ما در روش‌های Tucker به اجبار قدر مطلق پاسخ را ترسیم کردیم که مقادیر معنادار باشند که خود از عوامل خطأ می‌باشد.

سؤال ٣

(الف)











ب)











(ج)

هدف ما در تجزیه‌ی Tucker این است که با کاهش رتبه‌ی فرض شده برای بعد دوم که سایه است، بتوانیم فارغ از وضعیت سایه تصویر چهره‌ی اصلی و یکسانی را برای فرد مورد نظر حاصل کنیم. در نتیجه‌ی این هدف همانطور که مشاهده می‌شود تجزیه بسیار عالی عمل کرده و عملاً به جز اندکی تاثیر پذیری شدت نور تصویر حاصله از میزان سایه‌ی تصویر اصلی، چهره‌ی فرد با کیفیت بسیار بالایی بازسازی می‌گردد.

در تجزیه‌ی SVD اما بسته به رتبه‌ی در نظر گرفته شده برای تصویر هدف رفع نویز تصویر و فشرده سازی می‌باشد. در نتیجه با کاهش رتبه‌ی تجزیه، تصویر به صورت یکنواخت فشرده شده و جزئیات اش را از دست میدهد.

مشخصاً با توجه به هدف ما در استفاده از الگوریتمها، هر دو در راستای اهداف خود درست عمل می‌کنند. برای بازسازی تصویر اصلی فشرده سازی انجام شده توسط SVD بسیار بهتر بوده و جزئیات بیشتری مشابه با تصویر اصلی حفظ می‌شود. در طرف دیگر، تجزیه Tucker تصویر صورت بسیار با کیفیت بالایی را از تصویر سایه‌های مختلف جدا کرده و با این که برای بازسازی تصویر سایه‌دار اولیه مناسب نیست اما برای بازسازی تصویر اصلی بدون سایه‌ی چهره‌ی فرد بسیار مناسب می‌باشد.