

بە نام خداوے یکتا

گزارش تدریین دوھ دوھ دوھ سیسیتھ های تصمیم یار

نام و نام خانوادگی:

میین رمضانی 40311415016

نام استاد:

آقا دکتر امین هاشمی

آذر 1404

سوال تمرین :

شرح مسئله: یک دیتاست به فرمت CSV به همراه تمرین در اختیار شما قرار گرفته است که حاوی نظرات ۱۰۰۰ کاربر برای ۲۰۰ فیلم است. این نظرات در بازه ۱-۵ هستند و درایه های با مقدار صفر، درایه هایی هستند که کاربر نظری در مورد آنها نداده است. هدف پیش بینی مقادیر صفر در این دیتاست است. روش ALS را به طور کامل و دقیقا مطابق با مراحل ارائه شده در اسلایدهای درس پیاده سازی کنید.

در پایان مقادیر به دست آمده توسط روش ALS را گرد کنید. به این صورت که اگر مقدار اعشار کمتر از ۰.۵ بود به سمت پایین و در غیراینصورت به سمت بالا گرد کنید.

در پایان نیز خطای ماتریس پیش بینی شده با ماتریس اصلی را بر اساس Sum Square Error (SSE) گزارش کنید.

حل تمرین :

کدی که نوشته شده بصورت زیر هست :

```
import numpy as np
import pandas as pd

def SSE(R_true, R_pred_round):
    mask = (R_true != 0)
    diff = R_true[mask] - R_pred_round[mask]
    return np.sum(diff ** 2)

def ALS(R, k, lambda_reg, n_iters):
    np.random.seed(42)

    m, n = R.shape

    U = np.random.rand(m, k) * 5.0
    V = np.random.rand(n, k) * 5.0

    observed_mask = (R != 0)

    for it in range(n_iters):

        for i in range(m):
            idx_items = np.where(observed_mask[i, :])[0]
            if idx_items.size == 0:
                continue
```

```

V_i = V[idx_items, :]
r_i = R[i, idx_items]

A = V_i.T @ V_i + lambda_reg * np.eye(k)
b = V_i.T @ r_i

U[i, :] = np.linalg.solve(A, b)

for j in range(n):
    idx_users = np.where(observed_mask[:, j])[0]
    if idx_users.size == 0:
        continue

    U_j = U[idx_users, :]
    r_j = R[idx_users, j]

    A = U_j.T @ U_j + lambda_reg * np.eye(k)
    b = U_j.T @ r_j

    V[j, :] = np.linalg.solve(A, b)

R_hat = U @ V.T
R_hat_clipped = np.clip(R_hat, 1, 5)
R_hat_round = np.floor(R_hat_clipped + 0.5)
sse = SSE(R, R_hat_round)
print(f"Iteration {it+1}/{n_iters} => SSE : {sse:.2f}")

return U, V

df = pd.read_csv("dataset/matrix_1000x200_sparse40.csv")
R = df.values.astype(float)
print(f"Shape of R: {R.shape[0]}x{R.shape[1]}")

K = 70      # تعداد فاکتورهای نهان
LAMBDA = 0.1 # ضریب منظم سازی ( $\lambda$ )
N_ITERS = 20 # تعداد تکرار ALS

U, V = ALS(R, k=K, lambda_reg=LAMBDA, n_iters=N_ITERS)

R_hat = U @ V.T

R_hat_clipped = np.clip(R_hat, 1, 5)

R_hat_round = np.floor(R_hat_clipped + 0.5)

sse = SSE(R, R_hat_round)
print("Final SSE:", sse)

df_out = pd.DataFrame(R_hat_round)
df_out.to_excel("output/R_completed_round.xlsx", index=False)

```

توضیح و اجرای کد :

```
import numpy as np  
import pandas as pd
```

در ابتدای پیاده‌سازی، دو کتابخانه‌ی NumPy و Pandas با نام‌های مستعار np و pd فراخوانی شده‌اند. کتابخانه‌ی NumPy به عنوان ابزار اصلی محاسبات عددی و ماتریسی در پایتون مورد استفاده قرار می‌گیرد و عملیات‌هایی نظیر تولید مقادیر تصادفی، ضرب ماتریسی، اعمال توابع روی کل آرایه‌ها، و حل دستگاه‌های معادلات خطی را فراهم می‌کند. از آنجا که الگوریتم ALS مبتنی بر محاسبات خطی روی ماتریس‌های امتیازها و ماتریس‌های فاكتورهای نهان است، استفاده از NumPy برای پیاده‌سازی کارا و بهینه‌ی این محاسبات ضروری است.

کتابخانه‌ی Pandas نیز برای کار با داده‌های جدولی به کار گرفته شده است. در این پیاده‌سازی، Pandas وظیفه‌ی خواندن ماتریس امتیازها از فایل CSV و در انتهای ذخیره‌سازی ماتریس تکمیل شده در قالب فایل اکسل را بر عهده دارد. بدین ترتیب، NumPy نقش «هسته‌ی محاسبات ماتریسی» و Pandas نقش «مدیریت و ورود/خروج داده‌ها» را در پیاده‌سازی الگوریتم ALS ایفا می‌کنند.

```
def SSE(R_true, R_pred_round):  
    mask = (R_true != 0)  
    diff = R_true[mask] - R_pred_round[mask]  
    return np.sum(diff ** 2)
```

تابع SSE برای محاسبه‌ی مجموع مربعات خطأ (Sum of Squared Errors - SSE) بین ماتریس امتیازهای واقعی و ماتریس امتیازهای پیش‌بینی شده به کار می‌رود. در این تابع، ورودی R_true ماتریس امتیازهای اصلی (شامل مقادیر صفر به عنوان «عدم امتیازدهی») و R_pred_round ماتریس امتیازهای پیش‌بینی شده و گردشده است.

در قدم اول، با استفاده از عبارت mask = (R_true != 0) یک ماسک بولی ساخته می‌شود که در آن، تنها خانه‌هایی که امتیاز واقعی دارند (درایه‌های غیر صفر ماتریس اصلی) به صورت True علامت‌گذاری می‌شوند. بدین ترتیب، خطأ فقط روی داده‌های مشاهده شده محاسبه می‌شود و خانه‌هایی که در ماتریس اولیه برابر با صفر بوده‌اند (داده‌ی گمشده) در محاسبه‌ی خطأ لحاظ نمی‌گردند.

در ادامه، اختلاف بین مقادیر واقعی و مقادیر پیش‌بینی شده تنها در همین موقعیت‌های دارای امتیاز واقعی محاسبه می‌شود (diff = R_true[mask] - R_pred_round[mask]). سپس مربع این اختلاف‌ها گرفته شده و با استفاده از np.sum(diff ** 2) مجموع می‌شوند. خروجی تابع، مقدار نهایی SSE است که به عنوان معیار خطای کلی مدل روی داده‌های واقعی (غیر صفر) مورد استفاده قرار می‌گیرد.

```

def ALS(R, k, lambda_reg, n_iters):
    np.random.seed(42)

    m, n = R.shape

    U = np.random.rand(m, k) * 5.0
    V = np.random.rand(n, k) * 5.0

    observed_mask = (R != 0)

    for it in range(n_iters):

        for i in range(m):
            idx_items = np.where(observed_mask[i, :])[0]
            if idx_items.size == 0:
                continue

            V_i = V[idx_items, :]
            r_i = R[i, idx_items]

            A = V_i.T @ V_i + lambda_reg * np.eye(k)
            b = V_i.T @ r_i

            U[i, :] = np.linalg.solve(A, b)

        for j in range(n):
            idx_users = np.where(observed_mask[:, j])[0]
            if idx_users.size == 0:
                continue

            U_j = U[idx_users, :]
            r_j = R[idx_users, j]

            A = U_j.T @ U_j + lambda_reg * np.eye(k)
            b = U_j.T @ r_j

            V[j, :] = np.linalg.solve(A, b)

    R_hat = U @ V.T
    R_hat_clipped = np.clip(R_hat, 1, 5)
    R_hat_round = np.floor(R_hat_clipped + 0.5)
    sse = SSE(R, R_hat_round)
    print(f"Iteration {it+1}/{n_iters} => SSE : {sse:.2f}")

return U, V

```

تابع ALS پیاده‌سازی الگوریتم Alternating Least Squares برای فاکتور‌گیری ماتریس امتیازها است. ورودی این تابع شامل ماتریس امتیازها R، تعداد فاکتورهای نهان k، ضریب منظم‌سازی lambda_reg و تعداد تکرارهای الگوریتم

است. در ابتدا با استفاده از `np.random.seed(42)` بذر تولید اعداد تصادفی ثابت در نظر گرفته می‌شود تا نتایج قابل تکرار باشند. سپس ابعاد ماتریس امتیازها (تعداد کاربران و تعداد آیتم‌ها) استخراج شده و دو ماتریس U و V با ابعاد $m \times k$ و $n \times k$ به صورت تصادفی مقداردهی اولیه می‌شوند. این دو ماتریس نمایانگر بردارهای ویژگی پنهان کاربران و آیتم‌ها هستند. علاوه بر این، با استفاده از $\text{observed_mask} = (R != 0)$ ماسکی تعریف می‌شود که محل درایه‌های دارای امتیاز واقعی (غیرصفر) را مشخص می‌کند تا در بهروزرسانی‌ها فقط این داده‌های مشاهده شده در نظر گرفته شوند.

هسته‌ی اصلی الگوریتم در یک حلقه‌ی تکرار به تعداد n_iters اجرا می‌شود. در هر تکرار، ابتدا ماتریس U بهروزرسانی می‌شود؛ به این صورت که برای هر کاربر i ، ابتدا مجموعه‌ی آیتم‌هایی که آن کاربر به آن‌ها امتیاز داده است با استفاده از ماسک استخراج می‌گردد. در صورتی که کاربری هیچ امتیازی ثبت نکرده باشد، آن سطر نادیده گرفته می‌شود. سپس زیرماتریس V_i (شامل فاكتورهای آیتم‌های امتیاز داده شده) و بردار امتیازهای متناظر r_i از ماتریس R تشکیل می‌شوند. در ادامه، ماتریس

$$A = V_i^T V_i + \lambda I_k$$

و بردار

$$b = V_i^T r_i$$

محاسبه می‌شوند که متناظر با حل مسئله‌ی کمترین مربعات با منظم‌سازی Ridge هستند. در نهایت، با حل دستگاه خطی $AU_i^T = b$ به‌وسیله‌ی تابع `np.linalg.solve`، بردار فاكتورهای نهان کاربر i در سطر متناظر ماتریس U بهروزرسانی می‌شود.

در گام بعدی و در همان تکرار، ماتریس V به صورت مشابه بهروزرسانی می‌گردد. برای هر آیتم j ، مجموعه‌ی کاربرانی که به آن آیتم امتیاز داده‌اند تعیین می‌شود و در صورت فقدان امتیاز، آن آیتم نیز نادیده گرفته می‌شود. سپس زیرماتریس U_j (شامل بردارهای ویژگی کاربران امتیازدهنده) و بردار امتیازهای متناظر r_j استخراج می‌شوند. در ادامه، ماتریس

$$A = U_j^T U_j + \lambda I_k$$

و بردار

$$b = U_j^T r_j$$

محاسبه شده و با حل دستگاه $AV_j^T = b$ ، سطر متناظر آیتم j در ماتریس V بهروزرسانی می‌شود. بدین ترتیب، در هر تکرار، ابتدا بردارهای ویژگی کاربران و سپس بردارهای ویژگی آیتم‌ها به صورت متناوب (Alternating) و بر اساس داده‌های مشاهده شده بهروزرسانی می‌شوند.

پس از اتمام بهروزرسانی‌های U و V در هر تکرار، ابتدا ماتریس تقریبی امتیازها به صورت $R_{\text{hat}} = U @ V.T$ محاسبه می‌شود. سپس برای جلوگیری از خروج مقادیر پیش‌بینی شده از بازه‌ی مجاز امتیازدهی، این مقادیر در بازه‌ی [1, 5] برباره (clip) می‌شوند و با استفاده از عبارت $\text{np.floor}(R_{\text{hat_clipped}} + 0.5)$ به نزدیک‌ترین عدد صحیح طبق قانون گرد کردن مورد نظر مسئله تبدیل می‌گرددند. در ادامه، با فراخوانیتابع SSE، مقدار مجموع مربعات خطای ماتریس اصلی و ماتریس پیش‌بینی شده (فقط روی درایه‌های غیر صفر ماتریس اولیه) محاسبه شده و برای هر تکرار چاپ می‌شود تا روند همگرایی الگوریتم قابل مشاهده باشد. در پایان، تابع ALS دو ماتریس نهایی U و V را بازمی‌گرداند که از آن‌ها برای بازسازی ماتریس کامل‌شده امتیازها استفاده می‌شود.

```
df = pd.read_csv("dataset/matrix_1000x200_sparse40.csv")
R = df.values.astype(float)
print(f"Shape of R: {R.shape[0]}x{R.shape[1]}")
```

در خط اول، ماتریس امتیازها از روی فایل CSV خوانده می‌شود. برای این منظور از تابع `Pandas.read_csv` کتابخانه‌ی استفاده شده است. در خط دوم خروجی این دستور یک شیء از نوع `DataFrame` است که داده‌های جدول‌مانند فایل CSV (ردیف‌ها به عنوان کاربران و ستون‌ها به عنوان آیتم‌ها) را در خود نگه می‌دارد. در گام بعدی، به منظور انجام محاسبات ماتریسی با استفاده از `NumPy`، داده‌های این `DataFrame` به یک آرایه‌ی عددی تبدیل می‌شوند. بدین ترتیب ماتریس امتیازها به صورت یک آرایه‌ی دو بعدی از نوع `float` در متغیر `R` ذخیره می‌شود. در خط سوم ابعاد ماتریس (تعداد کاربران و تعداد آیتم‌ها) چاپ می‌شود تا مشخص گردد که اندازه‌ی داده‌های ورودی صحیح خوانده شده است.

```
K = 70      # تعداد فاکتورهای نهان
LAMBDA = 0.1 # ضریب منظم‌سازی
N_ITERS = 20 # تعداد تکرار ALS
```

در این بخش سه پارامتر اصلی الگوریتم تعیین شده‌اند. متغیر K تعداد فاکتورهای نهان (ابعاد فضای پنهان ویژگی‌ها) را مشخص می‌کند؛ هرچه مقدار K بزرگ‌تر انتخاب شود، مدل توانایی بیشتری برای مدل‌کردن الگوهای پیچیده‌ی سلیقه‌ی کاربران خواهد داشت، اما در مقابل احتمال بیش‌برازش (overfitting) و همچنین زمان محاسباتی افزایش می‌یابد. پارامتر (λ) ضریب منظم‌سازی است که در فرمول $ALS \lambda I_k$ افزوده می‌شود و با کنترل اندازه‌ی بردارهای U و V از بیش‌برازش جلوگیری می‌کند؛ مقدار بسیار کوچک λ مدل را مستعد بیش‌برازش و مقدار بسیار بزرگ آن باعث بیش‌از‌حد هموار شدن (underfitting) مدل می‌شود. پارامتر N_{ITERS} تعداد تکرارهای اجرای گام‌های متناوب بهروزرسانی U و V را تعیین می‌کند؛ افزایش این مقدار معمولاً تا جایی مفید است که مقدار خطای SSE (در تکرارهای متوالی تقریباً ثابت شود و پس از آن افزایش بیشتر تکرارها تأثیر قابل توجهی بر دقت ندارد).

```
U, V = ALS(R, k=K, lambda_reg=LAMBDA, n_iters=N_ITERS)
R_hat = U @ V.T
R_hat_clipped = np.clip(R_hat, 1, 5)
R_hat_round = np.floor(R_hat_clipped + 0.5)
```

در این بخش از برنامه ابتدا با فراخوانی تابع `ALS` و ارسال ماتریس امتیازها R به همراه مقادیر انتخاب شده برای تعداد فاکتورهای نهان (K)، ضریب منظم‌سازی (`lambda_reg`) و تعداد تکرارها (`n_iters`)، دو ماتریس فاکتورهای نهان

کاربران و آیتم‌ها به دست می‌آید؛ به این صورت که U نمایانگر بردارهای ویژگی پنهان کاربران و V نمایانگر بردارهای ویژگی پنهان آیتم‌ها است. پس از به دست آمدن این دو ماتریس، ماتریس تقریب‌زده امتیازها با ضرب ماتریسی U در ترانهادهی V ساخته می‌شود ($R_{\text{hat}} = U @ V.T$) که در واقع پیش‌بینی مدل از امتیاز هر کاربر به هر آیتم را ارائه می‌کند. از آنجا که مقادیر پیش‌بینی شده در R_{hat} ممکن است کمی از بازه‌ی معتبر امتیاز‌دهی (۱ تا ۵) خارج شوند، در گام بعدی با استفاده از تابع `np.clip` این مقادیر در بازه‌ی [۱, ۵] محدود می‌شوند تا همه‌ی پیش‌بینی‌ها در محدوده‌ی مجاز قرار گیرند و نتیجه در ماتریس $R_{\text{hat_clipped}}$ ذخیره می‌شود. در نهایت، برای تبدیل این مقادیر اعشاری به امتیاز‌های صحیح، از قاعده‌ی گرد کردن استفاده شده است؛ بدین صورت که با محاسبه‌ی `np.floor(R_{\text{hat_clipped}} + 0.5)` هر مقدار ابتدا $/5$ واحد افزایش یافته و سپس به پایین گرد می‌شود، بنابراین مقادیر با اعشار کمتر از $5/0$ به عدد پایین‌تر و مقادیر با اعشار برابر یا بیشتر از $5/0$ به عدد بالاتر نگاشت می‌شوند و ماتریس نهایی $R_{\text{hat_round}}$ شامل امتیاز‌های صحیح پیش‌بینی شده‌ی کاربران خواهد بود.

```
sse = SSE(R, R_hat_round)
print("Final SSE:", sse)
```

در این قسمت ابتدا با فراخوانی تابع `SSE`، مقدار مجموع مربعات خطای بین ماتریس امتیاز‌های واقعی و ماتریس امتیاز‌های پیش‌بینی شده محاسبه می‌شود. در این فراخوانی، R به عنوان ماتریس امتیاز‌های اولیه (که در آن فقط درایه‌های غیر صفر به عنوان داده‌های واقعی در نظر گرفته می‌شوند)، $R_{\text{hat_round}}$ به عنوان ماتریس امتیاز‌های نهایی پیش‌بینی و گردشده به تابع داده می‌شوند. تابع `SSE` همان‌طور که قبلًا توضیح داده شد، فقط روی خانه‌هایی که در ماتریس اصلی مقدار غیر صفر دارند اختلاف بین مقدار واقعی و مقدار پیش‌بینی شده را محاسبه کرده، مربع این اختلاف‌ها را به دست آورده و در نهایت مجموع آن‌ها را به عنوان یک عدد اسکالر برمی‌گرداند که نشان‌دهنده‌ی میزان خطای کلی مدل روی داده‌های مشاهده شده است. دستور `print("Final SSE:", sse)` نیز این مقدار را به عنوان خطای نهایی الگوریتم ALS بعد از اتمام همه‌ی تکرارها در خروجی نمایش می‌دهد تا عملکرد نهایی مدل از نظر دقیقت پیش‌بینی قابل ارزیابی باشد.

```
df_out = pd.DataFrame(R_hat_round)
df_out.to_excel("output/R_completed_round.xlsx", index=False)
```

در انتهای برنامه، پس از محاسبه‌ی ماتریس نهایی امتیاز‌های پیش‌بینی شده در قالب $R_{\text{hat_round}}$ لازم است این ماتریس به صورتی مناسب برای استفاده و تحلیل‌های بعدی ذخیره شود. برای این منظور، ابتدا با دستور `df_out = pd.DataFrame(R_hat_round)` تبدیل می‌شود تا امكان ذخیره‌سازی آن در قالب‌های مختلف فراهم گردد. سپس با استفاده از تابع `to_excel` این `DataFrame` در قالب یک فایل اکسل با نام `R_completed_round.xlsx` در مسیر `output` ذخیره می‌شود. در این فراخوانی، پارامتر `index=False` باعث می‌شود که اندیس سطرها (شماره‌ی ردیف‌های `DataFrame`) در فایل اکسل نوشته نشود؛ در نتیجه، فایل خروجی شامل یک ماتریس عددی خالص از امتیاز‌های پیش‌بینی شده است که هر سطر آن به جز سطر اول، متناظر با یک کاربر و هر ستون آن متناظر با یک آیتم می‌باشد و می‌توان آن را به طور مستقیم در محیط‌هایی مانند Excel مشاهده و تحلیل کرد.

خروجی نهایی کد بصورت زیر هست :

```
Shape of R: 1000x200
Iteration 1/20 => SSE : 97174.00
Iteration 2/20 => SSE : 80574.00
Iteration 3/20 => SSE : 72050.00
Iteration 4/20 => SSE : 67104.00
Iteration 5/20 => SSE : 63997.00
Iteration 6/20 => SSE : 61827.00
Iteration 7/20 => SSE : 59880.00
Iteration 8/20 => SSE : 58319.00
Iteration 9/20 => SSE : 57112.00
Iteration 10/20 => SSE : 56252.00
Iteration 11/20 => SSE : 55466.00
Iteration 12/20 => SSE : 54753.00
Iteration 13/20 => SSE : 53849.00
Iteration 14/20 => SSE : 53238.00
Iteration 15/20 => SSE : 52708.00
Iteration 16/20 => SSE : 52242.00
Iteration 17/20 => SSE : 51768.00
Iteration 18/20 => SSE : 51439.00
Iteration 19/20 => SSE : 51079.00
Iteration 20/20 => SSE : 50866.00
Final SSE: 50866.0
```

در این خروجی ابتدا دیده می شود که ماتریس ورودی با ابعاد 1000x200 به درستی خوانده شده و سپس مقدار SSE در طول 20 تکرار الگوریتم ALS گزارش شده است. در تکرار اول خطای حدود 97174 است و در هر تکرار به صورت نسبتاً یکنواخت کاهش پیدا می کند؛ به طوری که بعد از چند گام اول افت شدیدی دارد و از حدود تکرار 10 به بعد کاهش ها آرام و جزئی می شود (مثلا از حدود 56252 در تکرار 10 به حدود 50866 در تکرار 20 می رسد). این رفتار نشان می دهد که الگوریتم به خوبی در حال یادگیری ساختار داده است و با هر تکرار تقریب بهتری از ماتریس امتیازها به دست می آورد، اما پس از مدتی روند کاهش خطای به حالت تقریباً پایدار می رسد و افزایش تعداد تکرارها فقط بهبودهای کوچک ایجاد می کند. مقدار «Final SSE: 50866.0» نیز خطای نهایی مدل روی امتیازهای واقعی (فقط درایه های غیر صفر) را نشان می دهد و بیانگر این است که مدل هنوز کمی خطای دارد، اما نسبت به حالت اولیه به میزان قابل توجهی بهبود یافته است.

Thank You...