



Movie Recommending System

پروژه درس جبر خطی کاربردی

استاد درس:

دکتر ادیبی

اعضای گروه:

یسنا طالبی - ۴۰۰۳۶۱۳۰۴۵

ایلیا شعبان پور فولادی - ۴۰۰۳۶۱۳۰۴۱

Table of Contents

Data Manipulation.....	3
Calculating the SVD Matrices.....	4
List of Recommendations.....	5
SVD Functions	6
Implementation of Eigenvalue Calculator	7
Sample Output.....	8
References	9
Used Libraries	9

Data Manipulation

```
if __name__ == '__main__':
    ratings = pd.read_csv('ratings.csv')
    movies = pd.read_csv('movies.csv')

    user_id = int(input('Enter userId: '))

    user_movie = np.zeros((611, 9742))
    for i, index in enumerate(movies['movieId']):
        user_movie[0][i] = movies['movieId'][i]

    user_movie = pd.DataFrame(user_movie)
    copy = user_movie.copy()
    user_movie.columns = user_movie.iloc[0]

    for row in ratings.iterrows():
        user_movie.loc[int(row[1]['userId']),[row[1]['movieId']]] = row[1]['rating']
    user_movie.columns = copy.columns

    seen_movies = user_movie.columns[user_movie.iloc[user_id].to_numpy().nonzero()[0]]

    user_movie.drop(index=0, inplace=True)
    np_user_movie = user_movie.to_numpy()
```

در ابتدای برنامه داده های اولیه برنامه را میخوانیم و یک ماتریس 610×9742 عضوی به عنوان ماتریس کاربر فیلم درست میکنیم. سپس سطر صفرم این ماتریس را برای movieId متناسب با اندیس آن ها در ستون ماتریس ایجاد میکنیم.

ماتریس را به دیتافریم تبدیل کرده ایم و سپس عناوین ستون های این دیتافریم را movieId قرار داده ایم. سپس به تناسب سطر کاربر و ستون فیلم مرتبط امتیاز آن کاربر به فیلم را وارد میکنیم.

سپس فیلم هایی که کاربر گرفته شده در ورودی دیده است را در یک لیست ذخیره میکنیم. کاربر گرفته شده در ورودی کاربریست که میخواهیم top 5 movie recommended را برایش بدست بیاوریم.

سطر اول که movieId ها بود را حذف میکنیم و مجدد دیتافریم را به آرایه نامپای تبدیل میکنیم تا در ادامه برای svd از آن استفاده کنیم.

Calculating the SVD Matrices

```
V = calculate_v_transpose(np_user_movie)
U = calculate_u(np_user_movie)
sigma = calculate_sigma(np_user_movie)

"""
these lines are used to save the heavy calculated U S V matrices for later use

# np.save("V.npy", V)
# np.save("U.npy", U)
# np.save("sigma.npy", sigma)

# V = np.load("V.npy")
# U = np.load("U.npy")
# sigma = np.load("sigma.npy")

"""

u_k = U[:, :3]
sigma = np.diag(sigma)
sigma_k = sigma[:3, :3]
V_k = V[:, :3]

predicted_ratings = np.dot(np.dot(u_k, sigma_k), V_k)

similarities = cosine_similarity(predicted_ratings[user_id - 1, np.newaxis], predicted_ratings)

sorted_similarities = np.argsort(similarities[0])[::-1]
```

ماتریس های U , Σ و V را با توابع مختص محاسبه آنها حساب کرده ایم (شرح این توابع در ادامه آمده است). $**$ برای کاهش زمان هر بار اجرای برنامه ماتریس های بدست آمده را با تابع `save` نامپای ذخیره کرده ایم تا در اجرا های بعدی نیازی به محاسبه آنها نباشد. از ماتریس های بدست آمده و مقادیر مفید (مقادیری که بیشترین اطلاعات درباره ماتریس اصلی را به ما میدهند) آنها دوباره ماتریس کاربر فیلم را میسازیم. اینبار ماتریس بدست آمده شامل پیشبینی امتیاز هر کاربر به فیلم هایی که ندیده است نیز میشود. حال باید با استفاده از معیار شباهت `cosine similarity` مشابهت مقادیر پیشبینی شده هر کاربر را با دیگر فیلم ها بدست آوریم. سپس آنها را به ترتیب نزولی سورت کرده ایم.

List of Recommendations

```
movies_to_recommend = [movie_idx for movie_idx in range(len(predicted_ratings[user_id - 1])) if
                        movie_idx not in seen_movies]

recommendation_list = [movie_idx for movie_idx in sorted_similarities if movie_idx in movies_to_recommend]

num_recommendations = 5

recommendation_list = recommendation_list[:num_recommendations]

print(f'Recommendation List for User {user_id}:\n')
for row in recommendation_list:
    title = movies.iloc[row, 1]
    print(title)
```

در نهایت لیست فیلم های پیشنهادی طبق ماتریس پیشبینی و معیار شباهت cosine similarity

بدست می آید و ۵ تا فیلم برتر پیشنهادی را از آن استخراج میکنیم.

در نهایت این ۵ فیلم برتر به ترتیب چاپ میشوند.

SVD Functions

```
def calculate_u(M):
    B = np.dot(M, M.T)

    eigenvalues, eigenvectors = np.linalg.eigh(B)
    ncols = np.argsort(eigenvalues)[::-1]

    return eigenvectors[:, ncols]

1 usage
def calculate_v_transpose(M):
    B = np.dot(M.T, M)

    eigenvalues, eigenvectors = np.linalg.eigh(B)
    ncols = np.argsort(eigenvalues)[::-1]

    return eigenvectors[:, ncols].T

1 usage
def calculate_sigma(M):
    if np.size(np.dot(M, M.T)) > np.size(np.dot(M.T, M)):
        new_M = np.dot(M.T, M)
    else:
        new_M = np.dot(M, M.T)

    eigenvalues, eigenvectors = np.linalg.eigh(new_M)
    eigenvalues = np.sqrt(eigenvalues)
    return eigenvalues[::-1]
```

برای محاسبه ماتریس U ابتدا ضرب داخلی ماتریس اصلی در ترانپوز خود را بدست می آوریم. سپس بردارهای ویژه این ماتریس جدید که مربعی است را بدست می آوریم. و بردارهای این ماتریس را به صورت نزولی سورت میکنیم و سپس برمیگردانیم.

برای محاسبه V هم به همین شکل عمل کرده ایم. برای محاسبه σ بعد بزرگتر ماتریس را پیدا کرده و به صورت قبل عمل میکنیم با این تفاوت که مقادیر ویژه ماتریس بدست آمده را استفاده میکنیم.

Implementation of Eigenvalue Calculator

```
def eigen(matrix, num_iterations=100):
    n = len(matrix)
    eigenvalues = np.zeros(n, dtype=complex)
    eigenvectors = np.eye(n, dtype=complex)
    for _ in range(num_iterations):
        q, r = qr(matrix)
        matrix = np.dot(r, q)
    for i in range(n):
        eigenvalues[i] = matrix[i, i]
    return eigenvalues, eigenvectors

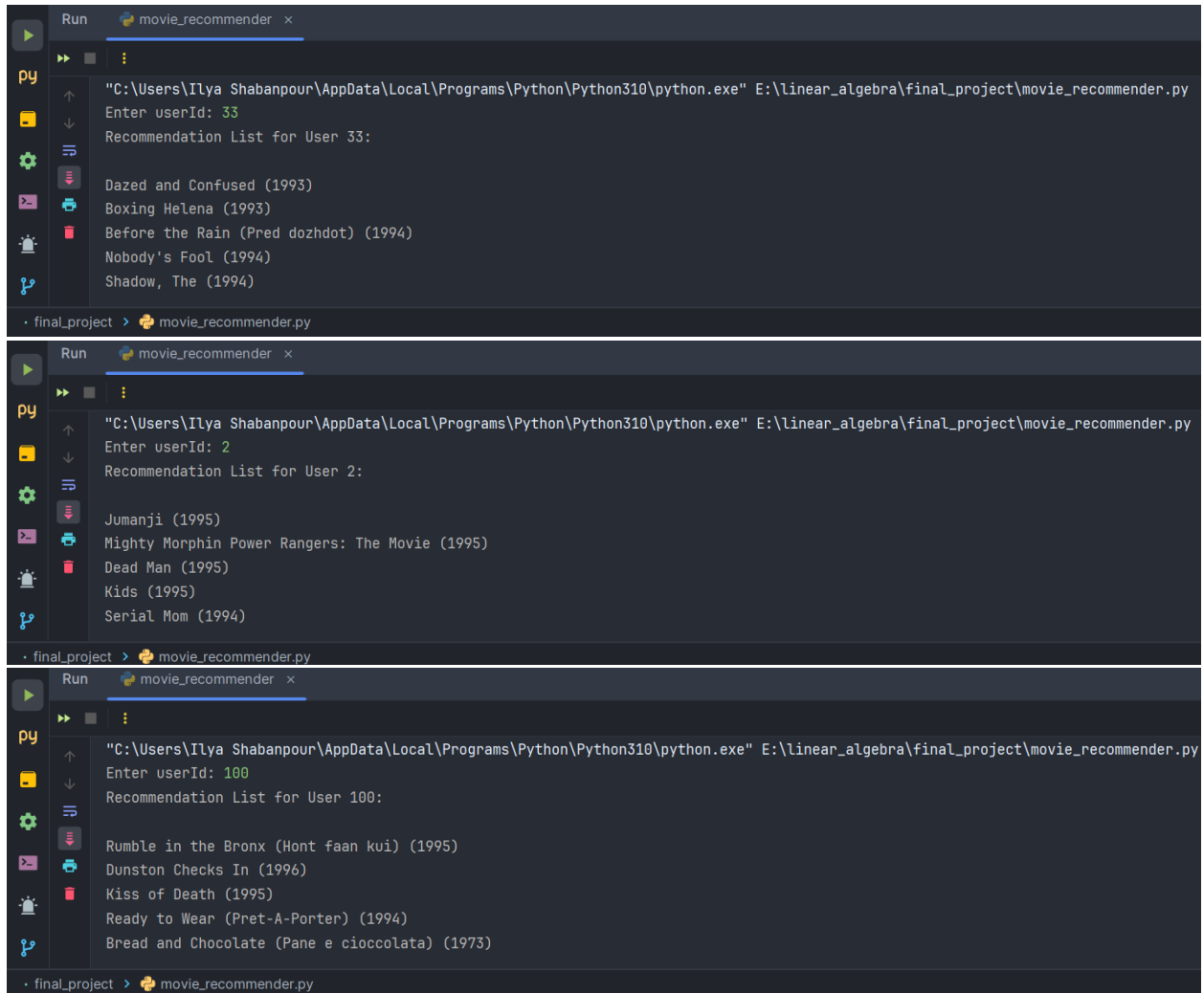
1 usage
def qr(matrix):
    n = len(matrix)
    q = np.zeros_like(matrix)
    r = np.zeros_like(matrix)

    for k in range(n):
        v = matrix[k:, k]
        norm_v = np.sqrt(np.sum(np.abs(v) ** 2))
        if norm_v == 0:
            q[k:, k] = 0
        else:
            q[k:, k] = v / norm_v
            r[k, k:] = norm_v * np.conj(q[k, k:])
            matrix[k:, k:] -= np.outer(q[k:, k], np.conj(r[k, k:]))
    return q, r
```

برای محاسبه بردارها و مقادیر ویژه به جای استفاده از توابع آماده نامپای توابع زیر را پیاده سازی کرده ایم. در تابع eigen با استفاده از الگوریتم تجزیه QR که در تابع qr پیاده سازی شده است، مقادیر و بردارهای ویژه بدست می آیند. این توابع برای ماتریس های بزرگ مثل ماتریس برنامه ما بسیار کند عمل میکنند به همین دلیل در تست اصلی برنامه این توابع استفاده نشده اند و توابع آماده نامپای را جایگزین کرده ایم.

Sample Output

برای سه کاربر: ۳۳، ۲ و ۱۰۰، ۵ فیلم برتر پیشنهادی را در خروجی نمایش داده ایم:



The image displays three sequential screenshots of a Python script's output in a terminal window. Each screenshot shows the execution of a program named 'movie_recommender.py' located at 'E:\linear_algebra\final_project\movie_recommender.py'. The script prompts the user to 'Enter userId:' and then displays a 'Recommendation List for User [userId]:'. The first screenshot shows the output for user 33, the second for user 2, and the third for user 100. The recommendations are listed with the movie title and year in parentheses.

```
Run movie_recommender x
>>>
"C:\Users\Ilya Shabanpour\AppData\Local\Programs\Python\Python310\python.exe" E:\linear_algebra\final_project\movie_recommender.py
Enter userId: 33
Recommendation List for User 33:
Dazed and Confused (1993)
Boxing Helena (1993)
Before the Rain (Pred dozhdot) (1994)
Nobody's Fool (1994)
Shadow, The (1994)

Run movie_recommender x
>>>
"C:\Users\Ilya Shabanpour\AppData\Local\Programs\Python\Python310\python.exe" E:\linear_algebra\final_project\movie_recommender.py
Enter userId: 2
Recommendation List for User 2:
Jumanji (1995)
Mighty Morphin Power Rangers: The Movie (1995)
Dead Man (1995)
Kids (1995)
Serial Mom (1994)

Run movie_recommender x
>>>
"C:\Users\Ilya Shabanpour\AppData\Local\Programs\Python\Python310\python.exe" E:\linear_algebra\final_project\movie_recommender.py
Enter userId: 100
Recommendation List for User 100:
Rumble in the Bronx (Hont faan kui) (1995)
Dunston Checks In (1996)
Kiss of Death (1995)
Ready to Wear (Pret-A-Porter) (1994)
Bread and Chocolate (Pane e cioccolata) (1973)
```


References

- <https://yeunun-choo.medium.com/singular-value-decomposition-in-a-movie-recommender-system-e3565ed42066>
- <https://medium.com/geekculture/singular-value-decomposition-calculation-using-eigenvalues-and-eigenvectors-in-python-dde785559174>
- ChatGPT

Used Libraries

Numpy

Pandas

Sklearn