liner algebra project

سهیل سلیمی-4003623018

مقدمه

تجزیه مقادیر منفرد یک روش ریاضی است که یک ماتریس را به سه ماتریس دیگر تجزیه می‌کند تجزیه مقادیر منفرد(SVD) می‌تواند در سیستم‌های توصیه‌گر یا ریکامندر سیستم‌ها کاربرد داشته باشد. این سیستم‌ها با تحلیل رفتار کاربران و محصولات، پیشنهادهایی را به کاربران می‌دهند که ممکن است برای آن‌ها جذاب باشند. SVD می‌تواند به سیستم‌های توصیه‌گر کمک کند که داده‌های ناقص یا پراکنده را پر کنند و عوامل مخفی را کشف کنند برای این پروژه، اگر یک ماتریس داشته باشیم که سطرهای آن کاربران و ستون‌های آن محصولات باشند و مقادیر آن امتیازاتی باشند که کاربران به محصولات داده‌اند، SVD می‌تواند این ماتریس را به سه ماتریس کوچک‌تر تجزیه کند که هر کدام از آن‌ها یک جنبه از رابطه بین کاربران و فیلم ها را نشان می‌دهند. بعد از این تجزیه، می‌توانیم با ضرب ماتریسی، مقادیر خالی را پر کنیم و بر اساس آن‌ها پیشنهادهایی را به کاربران بدهیم.

تابع norm

برای محاسبه نرم اقلیدوسی ماتریس از این تابع استفاده می کنیم.

تابع power_iteration

الگوریتم power_iteration یا روش توانی یک الگوریتم برای یافتن بزرگترین مقدار ویژه (`eigenvalue`) و بردار ویژه (`eigenvector`) متناظر یک ماتریس قابل تشکیل مجدد است این الگوریتم به شرح زیر است:

1. یک بردار تصادفی $x0$ با اندازه‌ی مساوی با تعداد سطرهای ماتریس A انتخاب می‌کنیم.
2. در هر تکرار k، بردار xk را با ضرب در ماتریس A به روز می‌کنیم: $xk+1=Axk$
3. بردار $xk+1$ را نرمال می‌کنیم: $xk+1=\| xk+1 \| xk+1$
4. اگر $xk+1$ و xk به اندازه‌ی کافی نزدیک به هم باشند، متوقف می‌شویم. در غیر این صورت، به مرحله‌ی 2 برمی‌گردیم.

بعد از پایان الگوریتم، بزرگترین مقدار ویژه تقریباً برابر با $xk+1Txk+1xk+1TAxk+1$ و بردار ویژه متناظر تقریباً برابر با $xk+1$ خواهد بود.

کد به صورت زیر خواهد بود

```
:def power_iteration(matrix, simulations=100)
Create a random initial vector #
rnd_vec = np.random.rand(matrix.shape[1])
:for _ in range(simulations)
Compute the matrix-by-vector product Ab #
```

```
m_b = np.dot(matrix, rnd_vec)
Compute the norm #
b_m_norm = norm(m_b)
Re-normalize the vector #
rnd_vec = m_b / b_m_norm
 Compute and return the largest eigenvalue and its corresponding #
eigenvector
return np.dot(np.dot(matrix, rnd_vec), rnd_vec) / np.dot(rnd_vec,
rnd_vec), rnd_vec
```

تابع eigen_values_vectors

روش اینجا از روش deflation که یک روش عددی است که با استفاده از یک مقدار ویژه و بردار ویژه شناخته شده، ماتریس را تبدیل به یک ماتریس کوچکتر می‌کند که مقادیر ویژه دیگر را دارد. این روش به ما اجازه می‌دهد که مقادیر ویژه را به ترتیب نزولی یا صعودی پیدا کنیم. برای انجام این کار، ما باید سه مرحله را طی کنیم:

- یک مقدار ویژه λ و بردار ویژه متناظر v را با استفاده از یک روش مانند روش توانی یا روش توانی معکوس پیدا کنیم.
- یک بردار ویژه چپ w را با حل کردن معادله $ATw = \lambda w$ پیدا کنیم، که در آن AT ماتریس ترانهاده A است.
- ماتریس A را با استفاده از فرمول $A1 = A - \lambda vwT$ به یک ماتریس کوچکتر $A1$ تبدیل کنیم، که مقادیر ویژه دیگر را دارد.

طرز کار کد به شکل زیر است:

- ابتدا اندازه آرایه را به عنوان n مشخص می‌کند.
- سپس دو آرایه صفر با اندازه (n, n) و (n) را به عنوان بردارها و مقادیر ویژه ایجاد می‌کند.
- برای هر شاخص از 0 تا n-1:
  - با استفاده از تابع `power_iteration`، بزرگترین مقدار ویژه و بردار ویژه متناظر آرایه را محاسبه می‌کند.
  - مقدار ویژه را در خانه‌ی شاخص از آرایه مقادیر ویژه قرار می‌دهد.
  - بردار ویژه را در ستون شاخص از آرایه بردارهای ویژه قرار می‌دهد.
  - آرایه را با کم کردن حاصل ضرب خارجی بردار ویژه و مقدار ویژه از آن به‌روز می‌کند. این کار باعث می‌شود که آرایه از مقدار ویژه بزرگترین خود کاسته شود و در تکرار بعدی مقدار ویژه بعدی را بیابد.
- در نهایت، آرایه‌های مقادیر ویژه و بردارهای ویژه را برمی‌گرداند.

```
:def eigen_values_vectors(matrix, simulations=100)
Get the size of the matrix #
n = matrix.shape[0]
Initialize the eigenvectors and eigenvalues with zero #
```

```
vectors = np.zeros((n, n))
values = np.zeros(n)
:for ind in range(n)
 Compute the largest eigenvalue and its corresponding eigenvector #
using power iteration
val, vec = power_iteration(matrix, simulations)
values[ind] = val
vectors[:, ind] = vec
matrix = matrix - val * np.outer(vec, vec)
return values, vectors
```

تابع SVD

این تابع با ورودی گرفتن یک ماتریس، ابتدا مقادیر و بردارهای ویژه ماتریس $matrix^T . matrix$ را محاسبه کرده، سپس، مقادیر ویژه را به صورت نزولی مرتب کرده و بردارهای ویژه متناظر با آنها را هم مرتب میکنیم. در نهایت، با توجه به فرمول تجزیه SVD مقادیر S ، U و TV را برمیگردانیم . البته از انجایی که در این پروژه نیازی به S و U نداریم ان ها حساب نمی کنیم.

```
:def SVD(matrix)
calculate eigen values and eigen vectors #
eigen_values, eigen_vectors = eigen_values_vectors(np.dot(matrix.T,
matrix))
sort eigen values and eigen vectors #
since we only need eigen_vectors for V_T we only sort them #
idx = eigen_values.argsort()[::-1]
eigen_values = eigen_values[idx] #
eigen_vectors = eigen_vectors[:, idx]
calculate SVD #
 in here we only calculate V_t since its the only thing we need for #
this project but other
prameters can be calculated too, as shown below #
S = np.sqrt(eigen_values) #
U = matrix.dot(V) / S #
V = eigen_vectors
return  V.T
```

تابع recommend

این تابع دو پارامتر `user_liked_movies_index` و `VT` می‌گیرد. این تابع قصد دارد یک لیست از شاخص‌های فیلم‌هایی را که بیشترین شباهت را به فیلم‌های مورد علاقه کاربر دارند، برگرداند.

- `user_liked_movies_index` که شاخص‌های فیلم‌های مورد علاقه `VT` اندیکسی برای آرایه کاربر را نشان می‌دهد.

- یک آرایه دو بعدی از اعداد حقیقی است که نماینده ماتریس ترانسفورماسیون می‌باشد. این `VT` ماتریس هر سطر آن مربوط به یک فیلم و هر ستون آن مربوط به یک ویژگی است. مقدار هر خانه نشان‌دهنده میزان داشتن آن ویژگی توسط آن فیلم است.

از این ماتریس برای محاسبه شباهت بین فیلم‌ها استفاده می شود. برای این کار، ابتدا یک لیست خالی به نام `rec` ایجاد می‌کند. سپس یک حلقه را از صفر تا تعداد ستون‌های ماتریس `VT` شروع می‌کند. در هر مرحله از حلقه، این کارها را انجام می‌دهد:

- یک متغیر به نام `i` را به عنوان شماره ستون در نظر می‌گیرد.
- یک متغیر به نام `column` را به عنوان ستون فعلی ماتریس `VT` در نظر می‌گیرد. این کار با استفاده از تابع `zip` انجام می‌شود که می‌تواند چندین آرایه را با هم ترکیب کند.
- یک متغیر به نام `similarity` را محاسبه می‌کند که برابر است با حاصلضرب نقطه‌ای بین ستون فعلی و ستون‌های مربوط به فیلم‌های مورد علاقه کاربر. این کار با استفاده از تابع `np.dot` انجام می‌شود که می‌تواند حاصل ضرب نقطه‌ای بین دو آرایه را برگرداند. این متغیر نشان‌دهنده میزان شباهت بین فیلم فعلی و فیلم‌های مورد علاقه کاربر است.
- یک زوج از شاخص فیلم فعلی و میزان شباهت آن را به لیست `rec` اضافه می‌کند.
- مقدار `i` را یک واحد افزایش می‌دهد.

پس از پایان حلقه، کد شما یک لیست از زوج‌های شاخص و شباهت را در اختیار دارد. سپس این لیست را بر اساس میزان شباهت مرتب می‌کند. این کار با استفاده از تابع `sorted` انجام می‌شود که می‌تواند یک لیست را بر اساس یک کلید مشخص مرتب کند. در اینجا کلید میزان شباهت است که در خانه دوم هر زوج قرار دارد. پارامتر `reverse=True` باعث می‌شود که لیست به صورت نزولی مرتب شود. در نهایت، فقط شاخص‌های فیلم‌ها را از لیست مرتب شده برمی‌گرداند.

```
:def recommend(user_liked_movies_index, VT)
[] = rec
i = 0
:for column in zip(*VT)
rec.append([i,np.dot(column,VT[user_liked_movies_index])])
i = i+1
final_rec = [i[0] for i in sorted(rec, key=lambda x: x[1],reverse=True)]
return final_rec
```

پیدا کردن توصیه برای کابران

- ابتدا دو فایل csv را با استفاده از تابع `pd.read_csv` می‌خواند. این دو فایل شامل داده‌های امتیازدهی کاربران به فیلم‌ها و اطلاعات فیلم‌ها هستند. مسیر این دو فایل در متغیرهای `PATH_RATING` و `PATH_MOVIE` ذخیره شده‌اند.
- سپس شماره کاربر مورد نظر را از طریق پارامترهای خط فرمان می‌گیرد. این کار با استفاده از تابع `int` و ماژول `sys` انجام می‌شود. شماره کاربر در متغیر `user_id` ذخیره می‌شود.
- بعد از آن، دو داده‌ست را با استفاده از تابع `pd.merge` با هم ادغام می‌کند. این کار با استفاده از ستون `movie_id` که در هر دو داده‌ست وجود دارد، انجام می‌شود. نتیجه این کار در

متغیر `merged` ذخیره می‌شود.

- سپس از داده‌ست ادغام شده یک ماتریس ایجاد می‌کند. این کار با استفاده از تابع `merged.pivot_table` انجام می‌شود. این تابع می‌تواند یک داده‌ست را به یک جدول خلاصه تبدیل کند. در اینجا، شاخص‌های سطرها شماره کاربران، شاخص‌های ستون‌ها عنوان فیلم‌ها و مقادیر خانه‌ها امتیازهای داده شده توسط کاربران به فیلم‌ها هستند. برای اینکه برنامه در زمان مناسب اجرا شود، فقط 610 کاربر اول و 500 فیلم اول را در نظر می‌گیرد. این کار با استفاده از تابع `iloc` انجام می‌شود که می‌تواند یک بخش از یک داده‌ست را برگرداند. همچنین، برای جایگزین کردن مقادیر نامعلوم با 0.1، از تابع `np.nan_to_num` استفاده می‌کند. ماتریس نهایی در متغیر `matrix` ذخیره می‌شود.
- سپس از روش SVD برای کاهش بعد ماتریس استفاده می‌کند. این کار با استفاده از تابع `SVD` انجام می‌شود که می‌تواند یک ماتریس را به سه ماتریس تجزیه کند. از این سه ماتریس، فقط ماتریس `V_t` که نماینده ویژگی‌های فیلم‌ها است، برای این پروژه لازم است. این ماتریس در متغیر `V_t` ذخیره می‌شود.
- سپس با استفاده از تابع `recommend` که قبلاً تعریف کرده‌اید، یک لیست از شاخص‌های فیلم‌هایی را که بیشترین شباهت را به فیلم‌های مورد علاقه کاربر دارند، برمی‌گرداند. این تابع دو پارامتر `user_id` و `V_t` را می‌گیرد و با محاسبه حاصلضرب نقطه‌ای بین ستون‌های ماتریس `V_t`، شباهت بین فیلم‌ها را می‌یابد. سپس این شباهت‌ها را به صورت نزولی مرتب می‌کند و فقط شاخص‌های فیلم‌ها را برمی‌گرداند. لیست نهایی در متغیر `recommendList` ذخیره می‌شود.
- سپس یک داده‌ست جدید با اطلاعات فیلم‌های پیشنهاد شده ایجاد می‌کند. این کار با استفاده از تابع `pd.DataFrame` و تابع `merge` انجام می‌شود. ابتدا یک داده‌ست با یک ستون به نام `movie_id` که شامل شاخص‌های فیلم‌های پیشنهاد شده است، ایجاد می‌کند. سپس این داده‌ست را با داده‌ست اطلاعات فیلم‌ها که در متغیر `movie` ذخیره شده است، ادغام می‌کند. این کار با استفاده از ستون `movie_id` که در هر دو داده‌ست وجود دارد، انجام می‌شود. داده‌ست نهایی در متغیر `df` ذخیره می‌شود.
- سپس داده‌ست نهایی را در یک فایل CSV ذخیره می‌کند. این کار با استفاده از تابع `df.to_csv` انجام می‌شود. نام فایل خروجی `output.csv` است.
- در نهایت، داده‌ست نهایی را با استفاده از تابع `print` چاپ می‌کند. همچنین یک پیام که می‌گوید که تمام داده‌ها در فایل `output.csv` قرار دارند، چاپ می‌کند.

```
read csv #
rating = pd.read_csv(PATH_RATING)
movie = pd.read_csv(PATH_MOVIE)
get the requested user as args #
user_id = int(sys.argv[1])
read the csv file and change it to matrix #
merged = pd.merge(rating, movie, on="movie_id")
 print("for this algoritem to run in reasonable amount of time we take only
first 500 moives but its can change to its original size. this's just faster
for testing \n")
for this algoritem to run in reasonable amount of time we take 20 user and #
```

```
 100 moives but its can change to its original size. this is just faster to
be tested
matrix = np.nan_to_num(merged.pivot_table(index="user_id", columns="title",
values="rating" ).iloc[:610 , :500] , nan= 0.1)
Compute the SVD(only the V_T part is need for this project) #
V_t = SVD(matrix)
find the recommend moive(sorted from best to worse to be liked) #
recommendList = recommend(user_id , V_t)
create data frame with moive info that has been recommend #
df = pd.DataFrame(data=recommendList , index=np.arange(len(recommendList))
,columns=['movie_id']).merge(movie, on="movie_id")
save the output to csv file#
df.to_csv("output.csv")
print(df)
print("all data is in output.csv")
```

بخشی از خروجی

> **Warning** ⚠
>
> همانطور که در بالا گفته شد برای پیدا کردن جواب در یک زمان معقول فقط 500 فیلم اول این دیتاست استفاده می شود. اگر نیاز دارید می توانید این مقدار را در کد تغییر دهید

بعد از اجرا `main.py 1` نتایج زیر برای کاربر شماره 1 به صورت زیر است

	movie_id	title	genres
0	23	Assassins (1995)	Action&Crime&Thriller
1	103	Unforgettable (1996)	Mystery&Sci-Fi&Thriller
2	178	Love & Human Remains (1993)	Comedy&Drama
3	460	Getting Even with Dad (1994)	Comedy
4	348	Bullets Over Broadway (1994)	Comedy
5	482	Killing Zoe (1994)	Crime&Drama&Thriller
6	282	Nell (1994)	Drama
7	339	While You Were Sleeping (1995)	Comedy&Romance
8	185	"Net, The (1995)"	Action&Crime&Thriller
9	493	Menace II Society	Action&Crime&Drama

	genres	title	movie_id
		(1993)	
10	Children&Comedy&Fantasy	Flintstones, The " "(1994)	355
11	Western	Wild Bill (1995)	210
12	Drama	Jefferson in Paris (1995)	254
13	Adventure&Comedy&Crime&Romance	Bottle Rocket (1996)	101
14	Children&Comedy	Heavyweights (Heavy Weights) (1995)	250
15	Drama&Romance	Immortal Beloved (1994)	249
16	Horror	Lord of Illusions (1995)	177

برای کاربر 20 به صورت زیر است

	genres	title	movie_id
0	Horror & Sci-Fi	Village of the Damned (1995)	332
1	Comedy & Drama	Ed Wood (1994)	235
2	Action & Thriller	"River Wild, The (1994)"	376
3	Drama & Musical	Farinelli: il castrato (1994)	242
4	Drama & Romance	Immortal Beloved (1994)	249
5	Comedy & Romance	Clueless (1995)	39
6	Action & Comedy & Crime & Drama & Thriller	Bad Boys (1995)	145
7	Comedy & Drama & Romance	I Like It Like That (1994)	359
8	Comedy & Drama	Unstrung Heroes (1995)	205
9	Action & Comedy & Crime & Thriller	Beverly Hills Cop III (1994)	420
10	Adventure & Drama	Lamerica (1994)	53
11	Comedy	"Birdcage, The (1996)"	141
12	Comedy	"Jerky Boys, The (1995)"	255
13	Children & Comedy & Fantasy	Gordy (1995)	243

	movie_id	title	genres
14	355	"Flintstones, The (1994)"	Children & Comedy & Fantasy
15	49	When Night Is Falling (1995)	Drama & Romance
16	274	Man of the House (1995)	Comedy

برای کاربر 280

	movie_id	title	genres
0	49	When Night Is Falling (1995)	Drama & Romance
1	242	Farinelli: il castrato (1994)	Drama & Musical
2	77	Nico Icon (1995)	Documentary
3	99	Heidi Fleiss: Hollywood Madam (1995)	Documentary
4	491	"Man Without a Face, The (1993)"	Drama
5	426	Body Snatchers (1993)	Horror & Sci-Fi & Thriller
6	383	Wyatt Earp (1994)	Western
7	405	Highlander III: The Sorcerer (a.k.a. Highlander: The Final Dimension) (1994)	Action & Fantasy
8	113	Before and After (1996)	Drama & Mystery
9	448	Fearless (1993)	Drama
10	332	Village of the Damned (1995)	Horror & Sci-Fi
11	18	Four Rooms (1995)	Comedy
12	349	Clear and Present Danger (1994)	Action & Crime & Drama & Thriller
13	282	Nell (1994)	Drama
14	423	Blown Away (1994)	Action & Thriller
15	249	Immortal Beloved (1994)	Drama & Romance
16	53	Lamerica (1994)	Adventure & Drama

منابع

power_iteration