گزارش ۶: ارائه ی یک سیستم توصیه گر برای یک مسئله ی کاربردی

نگارش

نیکا شهابی۹۷۱۳۰۲۳

استاد

دکتر مهدی قطعی

اردیبهشت ۱۴۰۰

لینک پروژه در گیتاب:

https://github.com/nikashahabi/collaborative-filtering-recommender-system

توضیح صورت مسئله:

امروزه با توجه به دیتای بسیاری که وب سایت ها و اپلیکیشن ها به آن دسترسی دارند این مسئله پیش آمده که چه دیتایی به کاربر نشان داده شود. به عبارتی چه مواردی به کاربر پیشنهاد داده شود. مثلا برنامه به کاربر پیشنهاد دهد که کجا غذا بخورد، چه محصولی بخرد، چه اهنگی گوش کند.. recommender system ها به برنامه برای انجام دادن این کار و دادن پیشنهادهایی مناسب با نیازهای کاربر کمک میکنند.این برنامه ها سعی میکنند pattern هایی پیدا کنند که با استفاده از آن ها پیشبینی کنند که کاربر به چه چیزی علاقه دارد.

دسته ای از recommender ها که rule-based هستند knowledge driven هستند. و دسته ای که از machine learning استفاده میکنند data driven هستند. در این پروژه از machine learning استفاده شده و برنامه data driven است. در knowledge-based recommender systems از یک kb و query دادن توسط کاربر به database استفاده میشود و اطلاعات قدیمی کاربر در ابتدا مورد توجه قرار نمیگیرد.

دو دسته از recommender system ها عبارتند از content-based و collaborative filtering (CF). در content-based از اطلاعات خود داده ها و شباهت آنها به همدیگر در دادن پیشنهاد به کاربر استفاده میکند. در مقابل در CF از شباهت کاربرها به همدیگر و سلایق کاربران مشابه برای پیشگویی استفاده میشود. CF میتواند یاد بگیرد که چه ویژگی هایی را مورد توجه قرار دهد و خود به دو قسمت تقسیم میشود: memory-based و model-based.

Memory-based CF خود دو مدل دارد: user-item و item-item. در user-item filtering برای یک کاربر مشخص کاربرهای شبیه به او پیدا میشوند و چیزهایی که آن کاربران دوست دارند پیدا میشود. در item-item filtering یک item مشخص انتخاب میشود. افرادی پیدا میشوند که از آن item خوششان می آید و سپس item های دیگری که آن افراد دوست دارند پیدا میشود.

در این پروژه memory-based CF (هر دو مدل آن) با استفاده از محاسبه ی cosine similarity پیشگویی را انجام میدهد و model-based CF با استفاده از پیاده سازی singular value decomposition (SVD).

توضیح درمورد دیتاست استفاده شده:

در این پروژه از movielens 100k dataset:

https://grouplens.org/datasets/movielens/100k

استفاده شده است. این دیتاست شامل ۱۰۰۰۰۰ rating(۱-۵) است که توسط ۹۴۳ user به ۱۶۸۲ تا فیلم داده شده است. u.data ی موجود فایلی است که شامل user_id، item_id، rating ، timestampاست. چند خط اول این فایل در شکل روبه رو نمایش داده شده است.

user_id	item_id	rating	timestamp
196	242	3	881250949
186	302	3	891717742
22	377	1	878887116
244	51	2	880606923
166	346	1	886397596

هر کاربر هم حداقل به ۲۰ تا فیلم امتیاز داده است.

در نتیجه این فایل شامل ۱۰۰۰۰۰ خط یا سطر و ۴ تا ستون است.

برای اطلاعات بیشتر درمورد این دیتابیس به لینک زیر مراجعه کنید.

http://files.grouplens.org/datasets/movielens/ml-100k-README.txt

توضیح درمورد مدل های استفاده شده و نحوه ی کار recommender system:

در ابتدا به وسیله ی sklearn.model_selection دیتاست داده شده با توجه به درصد داده شده به دوتا دیتاست training data set و testing data set تقسیم میشود. (shuffle + split)

```
trainData, testData = sc.train_test_split(df, test_size=0.25)
```

```
(75000, 4) shape of train
```

```
(25000, 4) shape of test
```

مراحل کار memory-based CF:

۱. برای train و test هر کدام یک ماتریس درست میکنیم که در درایه ی I,j آن rating ای که فردی با id عه I+1 به فیلمی با id عه j+1 داده است قرار گرفته است. هر کدام از این ماتریس ها را trainDataMatrix و testDataMatrix مینامیم.(۲۵ درصد داده ها را در test قرار میدهیم تا با استفاده از آن مدل را تست کنیم.)

trainDataMatrix = createDataMatrix(trainData, df)

testDataMatrix = createDataMatrix(testData, df)

۲. ماتریس similarity تولید میشود. برای CF item-item similarity از تمام کاربرانی استفاده میشود که به هر دو فیلم امتیاز داده اند. برای CF user-item similarity از تمام فیلم هایی که هر دو کاربر به آنها امتیاز داده اند استفاده میشود.

از sklearn استفاده میشود تا با استفاده از cosine similarity شباهت دو فیلم یا دو کاربر پیدا شود. برای شباهت دو کاربر از

```
userSimilarity = pairwise_distance(trainDataMatrix,
metric="cosine")
```

استفاده میشود و ماتریسی ۹۴۳*۹۴۳ تولید میشود (چون در training data set، ۹۴۳ تا کاربر داریم.)

همین طور برای یافتن شباهت دو item از

```
itemSimilarity = pairwise_distance(trainDataMatrix.T,
metric="cosine")
```

استفاده میشود که ۱۶۸۲*۱۶۸۲ است چون ۱۶۸۲ تا فیلم در training data set داریم.

حال توضیح میدهیم که cosine distance چه چیزی را محاسبه میکند. در حقیقت برای هر دو سطر از ماتریسی که به آن داده میشود فاصله را محاسبه میکند. مثلا فرض کنیم که trainDataMatrix با ابعاد (943, 1682) به آن داده شده. ماتریسی با ابعاد (943, 943) برمیگرداند که در درایه ی I, j و یا j, I آن فاصله ی دو سطر I ام و jام از ماتریس اولیه ی داده شده به آن (trainDataMatrix) قرار دارد. این شباهت هم برای سطر X و Y به این صورت محاسبه میشود که این دو سطر به عنوان دو بردار در نظر گرفته میشوند و شباهت X، Y به صورت normalized dot product ای از X, Y محاسبه میشود. (ضرب داخلی تقسیم بر اندازه ی X و Y)

$$K(X, Y) = \langle X, Y \rangle / (\|X\| * \|Y\|)$$

(برای اطلاعات بیشتر لینک های در منابع)
تفاوت و distance هم که در اینجا ما ذخیره اش میکنیم هست:

$$1 - K(X,Y)$$

۳. مرحله ی بعدی پیش بینی کردن با استفاده از similarity matrix هایی است که تولید کردیم. در این مرحله دو ماتریس که متناظر user-item و item-item collaboration filtering هستند تولید میشود که هر کدام ابعادی (943, 1682) دارند و در درایه ی I,j آن ها پیشگویی rating مناسب که کاربر I به فیلم j داده قرار گرفته است.

فرمول نوشته شده ی روبه رو برای محاسبه ی rating ای است که کاربر k به فیلم m میدهد. شباهت کاربر k با کاربران دیگر به عنوان وزن ضرب میشود تا کاربرانی که شباهت بیشتری دارند تاثیر بیشتری در نتیجه دارد. همچنین باید توجه شود که به جای فاصله ی مطلق rating ها از فاصله ی نسبی استفاده میشود تا اگر دو کاربر امتیازات مشابهی میدهند ولی به روش های مختلف امتیاز میدهند تفاوتی بینشان فرض نشود. (U بیانگر کاربر و x بیانگر rating)

$$\hat{x}_{k,m} = \bar{x}_k + \frac{\sum_{u_a} sim_u(u_k, u_a)(x_{a,m} - \bar{x_{u_a}})}{\sum_{u_a} |sim_u(u_k, u_a)|}$$

از فرمول رو به رو هم برای item-item prediction استفاده میشود.

$$\hat{x}_{k,m} = \frac{\sum_{i_b} sim_i(i_m, i_b)(x_{k,b})}{\sum_{i_b} |sim_i(i_m, i_b)|}$$

۴. حال نوبت این است که پیش بینی به دست آورده شده را به وسیله ی test data set تست کنیم. باید خانه هایی از ماتریس predict را در نظر بگیریم که در test set صفر نیستند. از RMSE استفاده میکنیم:

تابع mean_square_error از sklearn مجذور RMSE را میدهد.

$$RMSE = \sqrt{\frac{1}{N}\sum (x_i - \hat{x}_i)^2}$$

مراحل کار model-based CF:

این روش برای حالت هایی که ماتریس دیتای ما sparse و خالی است کاربرد بیشتری دارد. در این روش ویژگی های مهمی از فیلم ها و تمایلاتی از کاربران پیدا میشود و به وسیله ی آن پیشگویی ها انجام میشود.

۱. میزان خالی بودن ماتریس داده پیدا میشود:

```
sparsity = round(1.0 - len(df) / float(numberOfUsers(df) *
numberOfItems(df)), 3)
```

۲. از یک matrix-factorization مهم و معروف به نام singular value decomposition استفاده میشود. در حقیقت در این روش از تجزیه ی SVD استفاده میشود و بردار های در U همان ویژگی های مهم کاربران و بردار های در V ویژگی های مهم فیلم ها برای پیش بینی هستند.

$X = USV^T$

ماتریس های U و V متعامد هستند. ماتریس S ماتریس قطری ای است که درایه های رو قطر اصلی آن نامنفی هستند. (مقادیر تکین X) از تجزیه ی تقریبی X استفاده میشود تا پیش بینی ها انجام شود.

توضیح توابع مسئله:

در بین کد به صورت کامنت آورده شده است.

نمونه خروجی مسئله:

Users/nayvaka/Documents/University/Term6/AI/AIProjects/RecommenderSystem/venv/bin/python /Users/nayvaka/Documents/University/Term6//AI/AIProjects/RecommenderSystem/main.py

= user-item memory-based CF prediction

[0.4028866 0.40779982 0.41463415 ... 0.37072133 0.35859188 0.35042982]]

[0.10355706 0.10361595 0.10350982 ... 0.09838985 0.10310092 0.08999284]

[0.0776004 0.0768764 0.07674004 ... 0.07534507 0.07850169 0.07384924]

...

[0.04450465 0.04407591 0.04521118 ... 0.03872161 0.04030837 0.03127438]

[0.14817551 0.14664904 0.14812612 ... 0.14309169 0.13549469 0.1264598]

[[0.2537942 0.25245728 0.25996431 ... 0.22802514 0.20237531 0.20718321]

= item-item memory-based CF prediction

0.26403639 0.26155131 ... 0.43316477 0.55543976 1.57164739]]

[0.26372187

0.06328021- 0.0665692- ... 0.13070019 0.29637368   1.3219417]

[0.06319182-

0.09053- 0.09377484- ... 0.10814075 0.26640003 1.33524086]

[0.09032961-

...

0.1190448- 0.12197344- ... 0.06742091 0.22584233 1.19147549]

[0.11876815-

0.00970063- 0.01261992- ... 0.18447604 0.31785024 1.35866986]

[0.00939097-

0.11010835 0.10775899 ... 0.28036541 0.38609746 1.40263089]

[[0.11027122

the sparsity level is 93.7%

= model-based CF prediction

1.41933718e+00 2.31824318e+00 1.02098356e+00 ... 0.00000000e+00]]

[2.05472066e-02 7.96903668e-02-

1.20066420e+00 2.66122598e-01 9.42565517e-02 ... 0.00000000e+00]

[5.24915914e-03 -2.31635438e-02-

2.49018867e-01 -1.53330357e-01 8.09301111e-02 ... 0.00000000e+00]

[5.40097454e-04 -8.59332877e-03-

...

2.12496435e+00 -7.45275633e-02 2.72699924e-01 ... 0.00000000e+00]

[1.67592614e-02 -6.77372530e-03

4.42518344e-01 1.27575409e-02 -4.45951635e-01 ... 0.00000000e+00]

[4.28307737e-03 -2.51159799e-02

1.19514995e+00 1.81377065e+00 7.84221647e-01 ... 0.00000000e+00]

[[1.66785446e-02 2.65514847e-02

model-based CF evaluation with RMSE = 2.7167335143633355

user-item memory-based CF evaluation with RMSE = 3.4505141732855273

item-item memory-based CF evaluation with RMSE = 3.1207127211507006

Process finished with exit code 0

تحلیل خروجی:

هر چه RMSE بیشتر باشد نشان میدهد تفاوت مقادیر پیش بینی شده با مقادیر اصلی بیشتر است. طبق جدول زیر متوجه میشویم model-based CF بهتر عمل کرده است.

model-based CF	user-item memory-based CF	item-item memory-based CF	
2.7167335143633355	3.4505141732855273	3.1207127211507006	**RMSE**

پیشنهاد برای بهتر شدن پروژه:

هر دو مدل CF برای کاربری که تازه وارد شده است و یا ranking ای به فیلمی نداده دچار مشکل هستند. (مسئله ی cold start problem برای این دو قابل حل نیست.)

منابع:

https://github.com/zhangruiskyline/DeepLearning/blob/master/doc/Recommendation.md#collaborative-filtering

https://realpython.com/build-recommendation-engine-collaborative-/filtering

https://blog.cambridgespark.com/nowadays-recommender-systems-are-used-to-personalize-your-experience-on-the-web-telling-you-what-120f39b89c3c

برای ساختمان بندی اصلی پروژه و ایده برای CF استفاده شد.

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html#sklearn.metrics.pairwise.cosine_similarity

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_distances.html

از cosine similarity برای user-item و item-item similarity استفاده شد.

https://fairyonice.github.io/Learn-about-collaborative-filtering-and-weighted-alternating-least-square-with-tensorflow.html