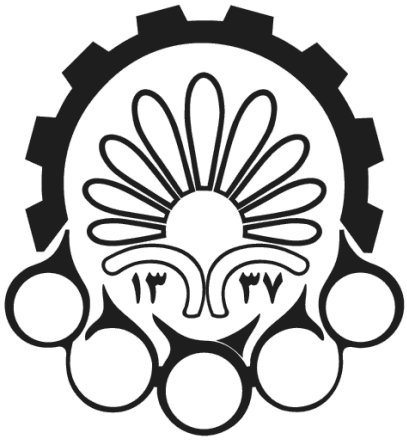
تمرین سری سه
درس جستجو و بازیابی اطلاعات در وب

استاد درس:
دکتر سعیده ممتازی

نام دانشجو:
آئیریا محمدی

شماره دانشجویی:
۴۰۲۱۳۱۰۲۸



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

ساختار فایل‌های پروژه به شکل زیر است:

ماجول (پوشه) mfactorization:

1. __init__.py
2. lrschedule.py
3. mf.py

ماجول (پوشه) util:

4. __init__.py
5. data_reader.py
6. evaluation.py
7. plot.py

پوشه اصلی (root):

8. bert_playground.ipynb
9. cv_plot.ipynb
10. visualization.ipynb
11. hyperparams.ipynb
12. hw_local.ipynb

توضیح مختصر هرکدام:
۱. حاوی تنظیمات ماجول است.

۲. حاوی تابع‌های است که در هر epoch یک learning rate مناسب برای الگوریتم matrix factorization ارائه دهد. به عنوان ورودی یک بازه می‌گیرد، ابتدا خیلی از وسطها به بیشترین مقدار می‌رود و سپس کم‌کم به مقدار کمینه میل می‌کند (و تغییرات آن به طور نمایی کم می‌شود).

۳. حاوی کد matrix factorization به کمک stochastic gradient descent. در این کد سعی شد برای حل بهینه، کل مسئله به شکل ماتریسی پیش برود. ابتدا تمام مقادیر ماتریس‌های p و q با هم آپدیت می‌شوند و در ادامه به شکل batch های کوچک‌تر تا یادگیری انجام شود. در مراحل پایانی اندازه batch به یک می‌رسد که معادل این است که مقدار یک نقطه از داده‌ها را در هر لحظه یاد می‌گیرد. همچنین در هر دور تنها از داده‌های مثبت (خانه‌های ماتریس با مقدار یک) نمونه تصادفی گرفته می‌شود و یادگرفته می‌شوند، چرا که مقادیر صفر به معنای عدم علاقه کاربر نیست بلکه صرفا یعنی تنها آن را ندیده است، پس ما نباید صفر بودن خانه‌های ماتریس را یاد بگیریم.

۴. تابع‌های مورد نیاز را import میکند و دیتاست را برای ماجول یک بار می‌خواند و کالاهای مورد علاقه هر کاربر را در داده‌های آموزش و آزمایش به یاد می‌سپارد (که در ارزیابی‌های مختلف این کار تکرار نشود).

۵. تابع‌های مورد نیاز برای خواندن مجموعه داده‌ها و اندیس‌گذاری کاربرها و آیتم‌ها. قابل توجه است که این اندیس‌گذاری باید برای هر دو مجموعه داده آموزش و آزمایش درست و مشترک باشد.

۶. پیاده‌سازی تابع‌های ارزیابی. ورودی آن ها ضرب دو ماتریس Q و P یا همان ماتریس تخمین زده شده ما از R است. قبل از ارزیابی، اول مطمئن می‌شویم که هیچ کدام از داده‌های آموزش در آزمایش نباشد، و سپس خروجی هر تابع ارزیابی را برای یکایک کاربران محاسبه می‌کنیم و در نهایت امتیازهای کاربران را میانگین می‌گیریم.

۷. ابزار کمکی برای ترسیم heatmap، که بتوان به شکل ملموس بازیابی‌های درست و غلط را روی ماتریس کاربر/آیتم آزمایش نشان داد.

۸. کدی که متناسب با خواسته‌های بخش سوم، کلمات بازخوردهای کالاها را کدگذاری می‌کند، از آن‌ها میانگین و max pooling می‌گیرد و ترکیبی خطی از آن‌ها را به عنوان بردار بازخورد معرفی می‌کند. میانگین بردارهای بازخوردهای هر کالا، بردار نماینده کالای ما خواهد بود.
این کد در colab در زمان ۵ دقیقه و ۷ ثانیه با مدل roberta و کارت گرافیک T4 اجرا شد.

۹. دفترچه‌ای برای آزمایش میزان یادگیری ماتریس توسط تابع mf (فاکتورگیری ماتریسی). به این شیوه عمل شد که همان ماتریس کالا/آیتم آموزش را به دو ماتریس با مقادیر ۰ و ۱ جدا کردیم و دومی را به عنوان مجموعه اعتبارسنجی استفاده کردیم (validation set). و در نهایت روند کاهش خطا برای مجموعه آزمایش جدید و مجموعه اعتبارسنجی را ترسیم کردم.

۱۰. تعداد توکن‌های هر جمله بازخورد کالاها را ترسیم کردم تا threshold مناسب برای تقطیع جمله‌های طولانی انتخاب کنم. مشاهده شد ۱۵ درصد داده‌ها طولی به مراتب بیشتر از سایر داده‌ها دارند، پس طول ۳۰۰ انتخاب شد که مقدار میانگین + انحراف معیار طول‌ها است.
۱۱. دفترچه‌ای شامل آزمایش‌های بی‌کران من برای پیدا کردن هایپرپارامترهای مناسب برای تابع فاکتورگیری. این دفترچه مرتب و خوانا نیست اما نمودارهای heatmap و scatter مختلفی دارد که برای مشاهده شخصی‌ام بود.

۱۲. دفترچه اصلی که به کمک ماجول‌ها و فایل‌های قبلی خواسته‌های این تمرین را به ترتیب برآورده می‌کند.
عملیات فاکتورگیری ماتریس کالا/آیتم در ۸۰۰ دور اجرا و اندازه نمونه ۵ درصد کل داده‌ها به مدت ۱۸ ثانیه به طول انجامید (با دستگاه m1 air). دلیل انتخاب این مقدارها تا حدی در دفترچه‌های cvplot و hyperparams نمایان است.

تحلیل نتیجه ارزیابی:

	method	recall	ndcg	rank correlation
0	R (train data)	3.847	0.007	1.000
1	R (test data)	99	1.000	1.000
2	matrix factorization	28	0.132	-0.006
3	BERT replace Q	2.262	0.004	0.010
4	BERT concat with Q	3.393	0.010	0.022

یک دلیل پایین بودن مقدار recall این است که ما تنها ۲۰ آیتم اول پیشنهادی هر کاربر را جدا می‌کنیم، در حالی که بعضی کاربرها در داده آزمایش، تا ۳۶ کالا را پسند کرده بودند (برای همین حتی ماتریس آزمایش هم مقدار کامل ۱۰۰ را نمی‌گیرد).

یک دلیل بالا بودن rank correlation برای ماتریس آموزش این است که ماتریس آموزش برای هیچ کدام از داده‌های ماتریس آزمایش مقداری متفاوت از صفر ندارد، و نسبت به تمامی آن کالاها بی‌تفاوت است، که با توجه به پیاده‌سازی انجام شده، بی‌تفاوت‌ها به ترتیب مشابه داده آزمایش چیده می‌شوند.
عکس پیاده‌سازی:

```python
def rank_correlation_fn(gold, scores):
    '''
    scores: user'th row of scoring matrix S (approximation for R)
    '''

    n = len(gold)
    a = np.arange(n)
    b = (scores[gold]*-1).argsort(kind='mergesort')

    if n == 1:
        return 1 if a[0] == b[0] else 0

    # spearman
    return 1 - (6*(b - a)**2 / (n * (n**2-1))).sum()
```

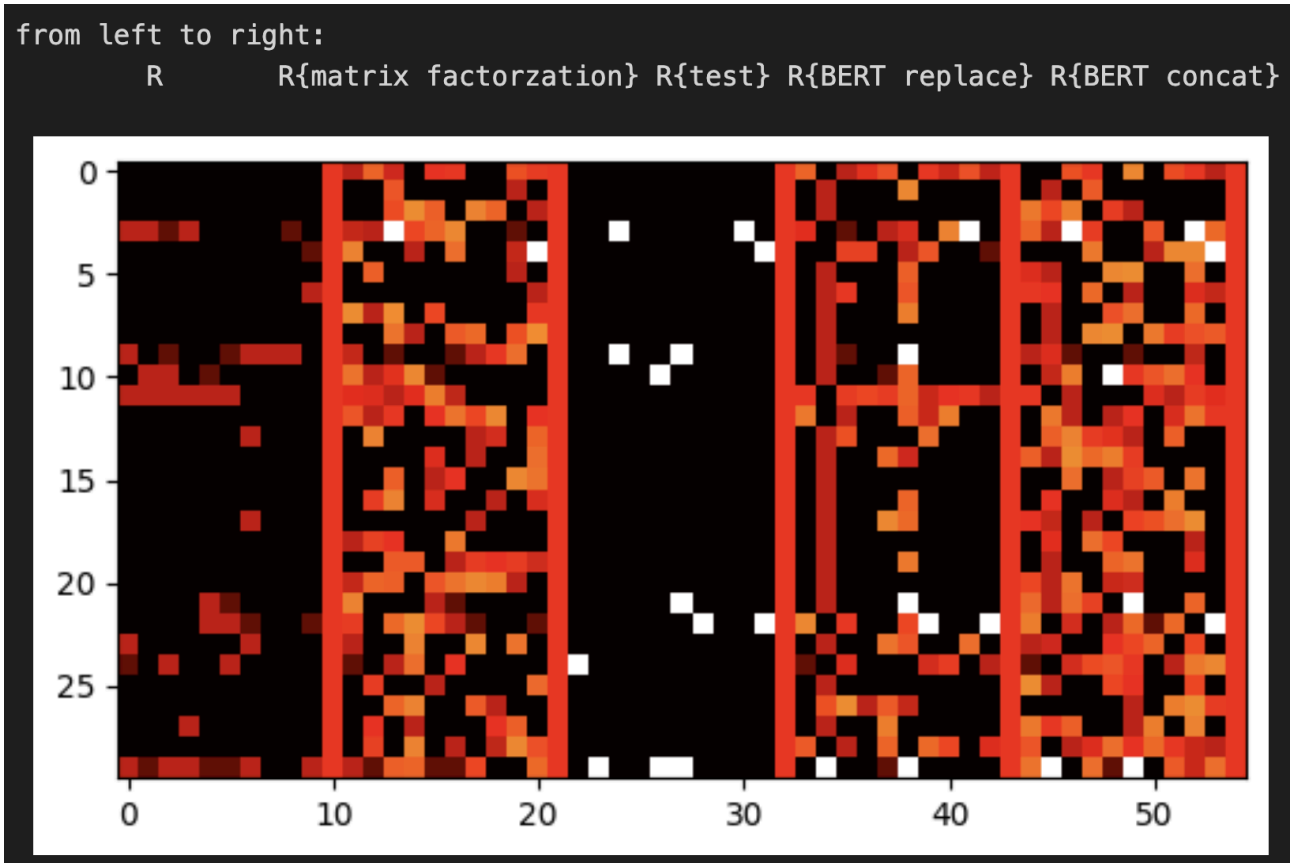
(نکته: مرتب‌سازی ادغامی، یک مرتب‌سازی پایدار است، و برای n=1 مخرج رابطه صفر می‌شد).

اگرچه مقدار بازیابی در هنگام استفاده از BERT زیاد نبود (ولی خیلی هم کم نبود)، ولی rank correlation قابل توجهی دارد و این مقدار وقتی ماتریس‌های Q و کدگذاری BERT هر هر دو یک نمایش فشرده از کالاها هستند (=بردارهای نماینده هر کالا زیر هم چیده شدند) استفاده بشود، همان‌طور که انتظار می‌رود نتیجه نسبت به حالت الف بهبود می‌یابد و rank correlation دوبرابر می‌شود.

در ادامه برای نمایش میزان کارایی الگوریتم‌ها، ماتریس‌های به دست آمده را در کنار ماتریس‌های اولیه مجموعه داده‌های آزمایش و آموزش قرار می‌دهیم و به شکل heat map ترسیم می‌کنیم. از آن‌جایی که ماتریس‌ها بزرگ بودند، تنها یک پنجره (زیر ماتریس) از آن‌ها را انتخاب می‌کنیم، به طوری که بیشترین مقدار کالا در داده آزمایش در آن پنجره با اندازه مشخص وجود داشته باشد.

توجه شما را به نقشه آخر جلب می‌کنم. رنگ قهوه‌ای معادل کالایی است که باید می‌یافتیم ولی مقدار ماتریس زیر threshold است (FN). رنگ سفید به معنای کالای درست یافت شده (TP, hit)، رنگ مشکی یعنی نه آن کالا در تست بوده و نه ما آن را گزارش کردیم (TN) ، و رنگ‌های دیگر بسته به میزان میل کردنشان به قهوه‌ای، میزان خطا را نشان می‌دهند برای کالایی که نباید گزارش می‌کردیم (FP).

از این نقشه نتیجه‌های زیر را می‌توان گرفت: (تصویر در صفحه بعد)



- هیچ کدام از داده‌های آموزش در این پنجره در بین داده‌های آزمایش قرار ندارند. تمامی نقاط داده آموزش تیره‌اند که یعنی در داده آزمایش مقدار صفر داشتند.
- داده‌های آزمایش با خودشان منطبق‌اند (نتیجه بدیهی)
- ماتریس به دست آمده در بخش ۲ (ماتریس دومی از سمت چپ)، نقاط رنگی مشابه با داده آموزش دارند (ماتریس چپ). این نشان‌دهنده این است که این ماتریس داده‌های آموزش را تا حد خوبی یاد گرفته است. اما از آن‌جایی که مقادیر آموزش در آزمایش قرار ندارند، گزارش مقدار بالا برای آن‌ها تعداد False positive ها را افزایش می‌دهد.
با این حال، دلیل این که با وجود تعدد نقاط تیره و خطا مقدار recall بالا برای matrix factorization گزارش کردیم این است که قبل ارزیابی، تمامی خانه‌های ماتریس که متناظر با داده آموزش هستند را کنار می‌گذاریم.
- دلیل این که ما در ماتریس ۲ نقاط hit زیاد نمی‌بینیم شاید این باشد که ما بر روی داده آموزش overfit کردیم، یا این که روش matrix factorization روش خیلی دقیقی برای تخمین سلیقه کاربرها نیست. و البته ما قطعا باید برای این ماتریس threshold متفاوتی برای نقاط retrieve شده در نظر می‌گرفتیم (چرا که نقاط با عدد خیلی بالا همان نقاط آموزش هستن، و ما باید آن‌ها را کنار بگذاریم و دنبال نقاط با مقدار متوسط باشیم).
- خانه‌های دو ماتریس راست از ماتریس‌های چپ روشن‌تر اند، و این نشان دهنده این است که لحاظ کردن معنای جملات توزیع مقادیر ماتریس را به سمت درستی میل می‌دهد (و با داده‌های آزمایش سازگارترند)

نکته بسیار مهم در نمایش بالا آن است که **تنها** در نمایش ما برای یک ماتریس M با مقدارهای بین صفر و یک (به طوری که در هر سطر/کاربر مقدار بیشینه یک و کمینه صفر باشد)، از مقدار threshold برابر با ۰.۵ استفاده کردیم (بیش از نیم = مرتبط، و زیر نیم = غیرمرتبط). با افزایش این عدد accuracy افزایش خواهد یافت و کاهش آن recall را می‌توان زیاد کرد و نقاط سفید بیشتری خواهیم داشت.
در زمان رتبه‌بندی و ارزیابی اما به threshold نیازی نبود چرا که ما نیاز بود ۲۰ آیتم اول را گزارش کنیم، مستقل از امتیاز آن‌ها، ولو کم باشد.