به نام خدا

داده کاوی

تمرین ششم

دکتر شاکری

محمد امین عرب خراسانی

۸۱۰۱۰۲۲۰۵

بهار ۱۴۰۳

بخش عملی

برای این بخش از google colab برای پیاده‌سازی خواسته‌های مسئله استفاده می‌شود. ابتدا دیتاست عنوان شده از لینک داده شده در فایل تمرین در google colab آپلود می‌شود و در ادامه به سوالات خواسته شده پاسخ داده می‌شود تا در نهایت الگوریتم خواسته شده روی این دیتاست پیاده‌سازی شود.

الف) فایل دیتاست از یک پوشه تشکیل شده است که اطلاعات مفید آن شامل ۲ فایل csv و یک فایل json می‌باشد. اطلاعات موجود در هر یک از این فایل‌ها به صورت جداگانه مورد بررسی قرار می‌گیرند.

فایل musae_git_edges.csv:

این فایل شامل دو ستون id_1 و id_2 می‌باشد که اتصال و ارتباط بین دو node در یک graph را نشان می‌دهد. در واقع شامل featureهای مربوط به edge می‌باشد. جدول زیر این فایل را در قالب یک دیتافریم نشان می‌دهد.

	id_1	id_2
0	0	23977
1	1	34526
2	1	2370
3	1	14683
4	1	29982
...	...	...
288998	37527	37596
288999	37529	37601
289000	37644	2347
289001	25879	2347
289002	25616	2347

289003 rows × 2 columns

فایل musae_git_target.csv:

این فایل شامل سه ستون id، name و ml_target می‌باشد که نام متعلق به هر id را نشان می‌دهد. همچنین ستون آخر مشخص می‌کند که آیا هر id توسعه‌دهنده‌ی یادگیری ماشین هست یا خیر. این فایل در قالب یک دیتافریم در زیر آورده شده است.

	id	name	ml_target
0	0	Eiryyy	0
1	1	shawflying	0
2	2	JpMCarrilho	1
3	3	SuhwanCha	0
4	4	sunilangadi2	1
...	...	...	...
37695	37695	shawnwanderson	1
37696	37696	kris-ipeh	0
37697	37697	qpautrat	0
37698	37698	Injabie3	1
37699	37699	caseycavanagh	0

37700 rows × 3 columns

فایل musae_git_features.json:

featureهای مربوط به nodeها در این فایل آورده شده است. فرمت این فایل json می‌باشد. بنابراین از کتابخانه‌ی json برای بررسی آن استفاده می‌شود. سپس اطلاعات موجود در آن مشاهده می‌شود تا برای بررسی بهتر، تصمیم مناسبی اتخاذ شود.

```
[('0',
  [1574,
   3773,
   3571,
   2672,
   2478,
   2534,
   3129,
   3077,
   1171,
   2045,
   1539,
   902,
   1532,
   2472,
   1122,
   2480,
   3098,
   2115,
   1578])]
```

به صورت کلی فایل json حاوی لیستی از featureها برای هر node می‌باشد. برای بررسی بهتر تعداد این featureها برای هر node استخراج می‌شود. پس از شمارش تعداد این featureها مشخص می‌شود که بیشترین تعداد feature در یک node برابر ۴۲ می‌باشد و همه‌ی nodeها این تعداد feature را لزوما ندارند.

برای ساده شدن کار با این ویژگی‌ها، برای هر کدام از featureها در هر node جداسازی انجام می‌شود تا در یک ستون در یک دیتافریم قرار بگیرند. نتیجه‌ی این تبدیل برای ۵ سطر ابتدایی در زیر آورده شده است.

	id	feature_0	feature_1	feature_2	feature_3	feature_4	feature_5	feature_6	feature_7	feature_8	...	feature_32	feature_33	feature_34	feature_35	feature_36	feature_37	feature_38	feature_39	feature_40	feature_41
0	0	1574	3773	3571	2672	2478	2534	3129	3077	1171.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	1	1193	376	73	290	3129	1852	3077	1171	1022.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2	1574	3773	925	1728	2815	2963	3077	364	1171.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	3	3964	3773	4003	928	1852	3077	364	1022	3763.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	4	1929	3773	1793	3511	1290	3129	3077	364	1171.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
37695	37695	1574	3773	73	1995	3554	1233	1789	345	3129.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
37696	37696	1929	3773	1663	1404	508	819	1852	3077	364.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
37697	37697	3433	3773	3104	1245	3129	1852	3077	364	1171.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
37698	37698	3730	3773	1695	2092	2954	1852	3077	364	1171.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
37699	37699	3433	509	1663	3437	676	3129	1852	3077	3763.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

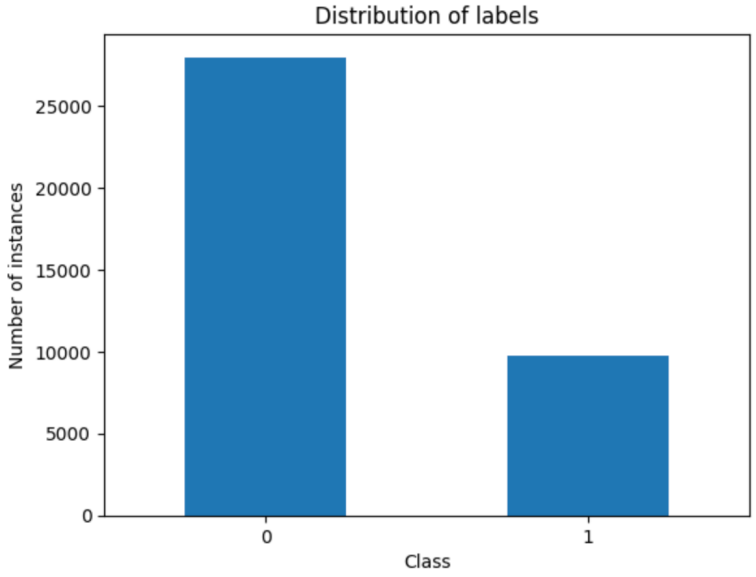
37700 rows × 43 columns

همانطور مشخص است، ۳۷۷۰۰ ردیف در این فایل موجود است که برای هر آیدی شامل featureهای متفاوت می‌باشد که می‌توان از آن‌ها برای اجرای الگوریتم استفاده کرد. (برای استفاده از این فایل نیازی به قالب دیتافریم نمی‌باشد)

ب) برای بررسی متعادل یا نامتعادل بودن این دیتاست می‌بایست تعداد لیبل‌های موجود در musae_git_target در هر دسته بررسی شود.

```
ml_target
0    27961
1     9739
Name: count, dtype: int64
```

همچنین برای نمایش بهتر این تعداد از histogram استفاده می‌شود. شکل زیر نتیجه‌ی شمارش این لیبل‌ها را نشان می‌دهد.



همانطور که از نتایج بالا مشخص است، این دیتاست نامتعادل می‌باشد. یکی از روش‌هایی که در تمرین قبل نیز برای متعادل کردن دیتاست از آن به کار برده شد، استفاده از SMOTE بود. در این دیتاست برای متعادل کردن دیتا نمی‌توان از این روش استفاده کرد چرا که با بالانس کردن فایل مربوط به لیبل نودها، ارتباط بین نودها که در فایل edge آورده شده است امکان ساختن ویژگی‌های جدید عملا وجود ندارد. (می‌توان از روش‌هایی مانند اضافه کردن ویژگی‌های مربوط به edgeها به صورت دستی یا با الگوریتم استفاده کرد)
به همین منظور روش جایگزین در نظر گرفته می‌شود. وزن مربوط به loss function کلاسی که تعداد کمتری دارد بیشتر در نظر گرفته می‌شود. این روش نامتعادل بودن دیتاست را از بین نمی‌برد اما باعث می‌شود که تاثیر نامتعادل بودن دیتاست بر روی آموزش مدل کاهش می‌یابد. به همین منظور در ادامه از CrossEntropyLoss استفاده می‌شود.

ج) در این قسمت ابتدا داده‌ی بررسی شده در قسمت قبل باید به صورتی تبدیل شود که بتوان آن را به عنوان ورودی به مدل GCNN داد. به همین منظور می‌بایست طول featureهای نودها هم‌اندازه شود. زیرا همانطور که اشاره شد این طول برای همه‌ی نودها یکسان نمی‌باشد. برای این منظور دو راه وجود دارد. روش اول استفاده از padding است به نحوی که featureهایی که مقادیر آن‌ها NaN می‌باشند با صفر پر می‌شوند. روش دوم استفاده از روش one-hot encoding می‌باشد. به این صورت که، به اندازه‌ی بیشترین مقدار در تمامی featureهای نودها، یک لیست تشکیل می‌شود که encoded_data می‌باشد و به این صورت است که اگر هر نود، یک مقدار مشخص را داشته باشد، ایندکس مربوط به آن مقدار ۱ و در غیر این صورت آن ایندکس ۰ در نظر گرفته می‌شود. این encoding برای تمامی نودها انجام می‌شود. بنابراین هر کدام از نودها شامل ۴۰۰۵ ویژگی می‌باشند که نشان‌دهنده‌ی وجود یا عدم وجود یک مقدار مشخص از ویژگی‌ها می‌باشد.
حال که طول این featureها برابر شده است، می‌بایست از torch استفاده کرد تا دیتا برای استفاده از مدل GCNN آماده شود. به همین منظور از torch_geometric.data استفاده می‌شود. ورودی مدل پس از استفاده از این کتابخانه به شکل زیر خواهد بود که در آن x مربوط به featureهای نودها، edge_index مربوط به featureهای egdeها و y لیبل هر نود می‌باشد.

```
Data(x=[37700, 4005], edge_index=[2, 289003], y=[37700])
```

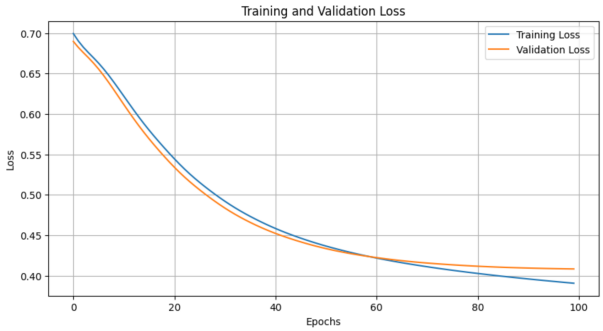
در ادامه معماری مربوط به مدل تعریف می‌شود. ابتدا از کتابخانه‌ی torch.nn استفاده می‌شود که پایه‌ی تمامی مدل‌های شبکه عصبی می‌باشد. در ادامه دو لایه تعریف می‌شود. لایه‌ی اول به تعداد num_features ورودی

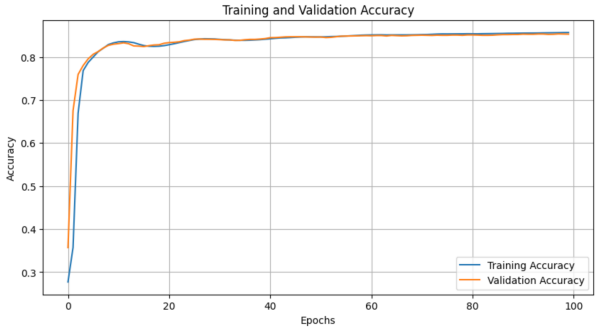
می‌گیرد و ۱۶ feature را به عنوان خروجی لایه‌ی اول بر می‌گرداند. لایه‌ی دوم نیز ۱۶ feature ورودی می‌گیرد و به تعداد کلاس‌های موجود خروجی می‌دهد که در این مثال لیبل‌ها ۰ یا ۱ می‌باشد. بنابراین خروجی این لایه ۲ می‌باشد که مشخص می‌کند آیا یک id می‌تواند توسعه‌دهنده‌ی یادگیری ماشین باشد یا خیر.

در ادامه mask ساخته می‌شود که کد آن در فایل ضمیمه قرار داده شده است. ۸۰ درصد دیتاست مختص فرآیند یادگیری، ۱۰ درصد برای تست و ۱۰ درصد آخر برای validation در نظر گرفته شده است.

همچنین وزن‌های مربوط به کلاس ۱ برای loss function نیز در این قسمت تنظیم شده است. برای آموزش مدل از optimizer ADAM استفاده شده و learning rate برابر ۰.۰۰۱ می‌باشد.

نتیجه‌ی آموزش مدل برای داده‌های آموزش و validation در دو نمودار مجزا در زیر آورده شده‌اند.





د) برای ارزیابی مدل ارائه شده روی داده‌های تست از مقادیر loss، accuracy، precision، recall و f1 score استفاده می‌شود. کد مربوط به این قسمت در فایل ارسالی ضمیمه شده است. نتیجه‌ی بررسی این پارامترها در زیر آورده شده است.

```
Test Loss: 0.4462
Test Accuracy: 0.8289
Test Precision: 0.7656
Test Recall: 0.8018
Test F1 Score: 0.7798
```

بخش تشریحی

الف) به کمک شبکه‌ی عصبی گرافی می‌توان ارتباط‌های پیچیده بین کاربران و محصولات را استخراج کرد.

به کمک شبکه‌های عصبی می‌توان پترن‌های موجود در رفتار مصرف‌کنندگان را شناسایی کرد، نمایش‌های پیچیده و غیرخطی از داده‌ها را به دست آورد تا از آنها برای پیش‌بینی رفتارهای کاربران استفاده کرد. معمولا رفتار کاربران در مواردی مانند خرید یک محصول بسیار پیچیده است. به طوری که به کمک شبکه‌های عصبی به خصوص یادگیری عمیق می‌توان الگوهای این رفتارها را از دیتاست شناسایی کرد. همچنین این رفتارها به دلیل پیچیدگی بالایی که دارند منجر به ویژگی‌های غیرخطی می‌شوند که استخراج آن‌ها به کمک شبکه‌های عصبی میسر است. شبکه‌های عصبی با استفاده از Activision functionهای غیر خطی می‌توانند ارتباط بین ویژگی‌های ورودی و خروجی را به خوبی شناسایی کنند تا یک سیستم recommender بهتر ارائه دهند.

از طرفی گراف‌ها راه قدرتمندی برای مدل‌سازی و ترکیب اطلاعات در ارتباط بین کاربر و محصول ارائه می‌دهند و به سیستم اجازه می‌دهند تا روابط مستقیم و غیرمستقیم را در نظر بگیرد. در این سوال هر کدام از محصولات و کاربران می‌توانند یک نود باشند که featureهای خاص خود را دارند که به واسطه‌ی edgeها با یکدیگر ارتباط دارند. Featureهای مربوط به این edgeها شامل اطلاعاتی مانند مشاهده‌ی محصول، خرید و امتیازدهی می‌باشد. گراف‌ها می‌توانند تمامی اطلاعات را در قالب‌های مختلف ذخیره کنند که این باعث عملکرد بهتر آن‌ها می‌شود. چنین ویژگی‌ای در این گراف‌ها منجر به آن می‌شود که نودها حتی اگر به صورت مستقیم نیز با یکدیگر در تعامل نباشند می‌توانند بر یکدیگر بر اساس میزان نزدیک بودن featureهایشان اثر بگذارند.

با توجه به تمامی موارد اشاره شده استفاده از شبکه‌های عصبی گرافی انتخاب مناسبی برای طراحی یک سیستم recommender می‌باشد.

ب) برای آن که بتوان از یک گراف به عنوان ورودی یک شبکه‌ی عصبی استفاده کرد، باید توجه داشت که نیاز به داشتن نودها، edgeها و ویژگی‌های مربوط به هر کدام از آن‌ها می‌باشد. در واقع در مرحله‌ی اول ساختار گراف مشخص می‌شود.

برای مرحله‌ی دوم، با توجه به هدف سوال، نودهای این گراف از کاربران و محصولات تشکیل می‌شوند. هر کدام از این نودها می‌توانند بر اساس فعالیتی که انجام داده‌اند/روی آن‌ها انجام شده است، featureهای مربوط به خود را داشته باشند. برای مثال، featureهای مربوط به کاربران می‌تواند دموگرافیک، رفتار گذشته و... باشد. همچنین featureهای محصولات می‌تواند شامل مواردی نظیر دسته‌بندی، قیمت، توضیحات و... باشد.

در مرحله‌ی بعدی featureهای مربوط به edgeها مشخص می‌شوند. این ویژگی‌ها می‌توانند شامل زمان تعامل، مشاهده‌ی محصول و مواردی از این دست باشد.

در مرحله‌ی آخر مولفه‌های اصلی گراف بر اساس مراحل قبل ایجاد می‌شوند و به عنوان ورودی از آن‌ها استفاده می‌شود. adjacency matrx و edge list برای استفاده به عنوان ورودی شبکه‌ی عصبی مناسب هستند. با استفاده از adjacency matrx در واقع ارتباط بین هر نود به همراه ویژگی‌های موجود در هر edge، در یک ماتریس قرار می‌گیرد که می‌توان از آن برای تعریف گراف استفاده کرد. این لینک نحوه‌ی پیشرفت یک adjacency matrx را تا کامل شدن آن برای یک مثال خاص به خوبی تشریح می‌کند.

ج) استفاده از شبکه‌های عصبی گرافی، این امکان را می‌دهد که به کمک مکانیزم message-passing برای هر کدام از نودها embedding انجام شود. مکانیزم message-passing به آن صورت است که در هر تکرار، هر نود اطلاعاتی را از همسایگی خود جمع‌آوری می‌کند و با پیشرفت این تکرارها، هر نود که embedding شده حاوی اطلاعات بیشتر و بیشتری از محدوده‌های بیشتر نمودار است. همین امر منجر به آن می‌شود که سیستم recommender طراحی شده با این روش، کارایی بیشتری داشته باشد. در واقع با این روش محصولات مشابه و کاربران مشابه شناسایی می‌شوند و همین دلیل کمک می‌کند که سیستم recommender بهتری ساخته شود.

د) برای این قسمت سه چالش اصلی به همراه راه‌حل احتمالی آن‌ها آورده شده است.

چالش اول: به دلیل بالا بودن تعداد نودها هم برای کاربران و هم برای محصولات، ساخت یک سیستم recommender بسیار هزینه‌ی محاسباتی بالایی دارد. به منظور کمتر کردن این هزینه‌ها، می‌توان از روش‌هایی مثل mini-batch training و graph sampling استفاده کرد.

چالش دوم: ویژگی‌های مربوط به محصولات، کاربران و حتی edgeها به صورت پیوسته تغییر می‌کنند. پیاده‌سازی یک incremental learning یا یک periodic retraining می‌تواند این چالش را برطرف کند.

چالش سوم: محصولات و کاربران جدید، اطلاعات زیادی ندارند که سیستم recommender بتواند برای آن‌ها recommend مناسبی داشته باشد. استفاده از transfer learning می‌تواند ایده‌ی مناسبی برای حل این مسائل باشد.