بررسی الگوریتم‌های Deepwalk و Node2Vec در سامانه‌های توصیه‌گر

دانشگاه الزهرا

استاد راهنما: سرکار خانم دکتر بیتا شمس

مرداد ماه ۱۳۹۹

فهرست

۱ مقدمه

۲ الگوریتم‌ها

۳ تعاریف

۴ ارزیابی

مقدمه

سامانه‌ها یا موتورهای توصیه‌گر با بررسی و تحلیل رفتار کاربران مناسب‌ترین اقلام چون فیلم، کالا و... را به آن‌ها پیشنهاد می‌کنند. حجم روزافزون اطلاعات، توجه هرچه بیشتر پژوهشگران را به این سامانه‌ها معطوف داشته است تا با ارائه الگوریتم‌هایی بهینه‌تر بتوانند در دستیابی کاربران به هدفشان آن‌ها را یاری کنند. گراف‌ها که شبکه‌های اجتماعی، خطوط حمل و نقل و... نمایندگان آن‌ها در زندگی حقیقی هستند به کمک پژوهشگران در این زمینه شتافته‌اند.

مدل‌سازی به وسیله گراف به پژوهشگران کمک می‌کند تا به درک بهتری از کاربران، محصولات و روابط بین آن‌ها در یک سامانه دست یابند. کاربران و محصولات، خود هرکدام نماینده یک گره منحصر به فرد در گراف متناظر با سامانه هستند و روش‌های متعددی تا به امروز برای بررسی ساختار حاکم بر روابط میان آن‌ها ارائه شده است. به طور کلی این روش‌ها در چهار دسته مختلف جای می‌گیرند که عبارت‌اند از:

۱. Node Classification
۲. Link Prediction
۳. Clustering
۴. Visulization

روش شماره ۱ درصدد یافتن برچسب گره‌ها با توجه به برچسب گره‌های موجود و توپولوژی شبکه است. روش شماره ۲ درصدد یافتن روابطی بین گره‌ها است که می‌توانند در آینده رخ دهند. روش شماره ۳ درصدد یافتن گره‌های مشابه و ادغام آن‌ها به عنوان یک گروه است و در انتها روش شماره ۴ درصدد فراهم آوردن بینشی برای فهم ساختار حاکم بر شبکه است. در چند دهه اخیر پژوهش‌های زیادی در این زمینه شکل گرفته است اما آنچه که به تازگی توجه محققان را به خود جلب کرده است الگوریتم‌هایی است که از نمایش گره‌های گراف در فضایی برداری استفاده می‌کنند. تلاش های صورت گرفته در بهبود الگوریتم‌های موجود در روش اول سبب ایجاد الگوریتم‌هایی با عنوان Random Walk گشته است که از جمله الگوریتم‌های نامبرده هستند.

الگوریتم‌ها

همانطور که گفته شد الگوریتم‌های Random Walk الگوریتم‌هایی هستند که با استفاده از نمایش گره‌های گراف در فضایی برداری سعی در پیشنهاد مناسب‌ترین اقلام به کاربران را دارند. این الگوریتم‌ها زمانی به کار می‌آیند که شبکه به قدری بزرگ است که نمی‌توان به طور کامل آن را ذخیره نمود و تنها امکان مشاهده بخشی از آن فراهم است. اساس کار این الگوریتم‌ها یافتن مسیرهایی تصادفی با شروع از یک گره و انتخاب گره‌ی بعدی با احتمالی معین است.

دو نوع از مهم‌ترین و معروف‌ترین انواع این الگوریتم ها عبارت‌اند از:

Deepwalk

Node2Vec

الگوریتم Deepwalk

گراف G=(V, E) با مجموعه رئوس V و مجموعه یال‌های E را در نظر بگیرید. در شروع، الگوریتم درختی دودویی از G می‌سازد و با انتخاب راسی چون W_i از مجموعه رئوس به طور تصادفی روند را ادامه می‌دهد. راس بعدی در این مسیر تصادفی از میان رئوس همسایه W_i انتخاب می‌گردد و الگوریتم خود با توجه به آرگومان‌هایی مختلف بیشترین احتمال را برای انتخاب هر یک از رئوس همسایه محاسبه می‌کند. هدف نمایش هر گره از بردار در فضای برداری R^d است بنابراین به ازای هر گره در گراف، مسیری تصادفی توسط الگوریتم طی می‌شود و در هر مرحله نمایش گراف به‌روز می‌شود.

الگوریتم Node2Vec

این الگوریتم بسیار شبیه به الگوریتم قبلی عمل می‌کند. تفاوت عمده آن با الگوریتم قبلی در بکارگیری استراتژی‌های کلاسیک جستجو یعنی دو الگوریتم جستجوی اول سطح و جستجوی اول عمق در انتخاب رئوس مسیر تصادفی است. همین امر سبب می‌شود تا بردار متناظر با هر گره در گراف از کیفیتی بالاتر برخوردار باشد و جزییات اطلاعاتی بیشتری از گراف را در اختیار داشته باشد.

در پایان هر دو الگوریتم با استفاده از Stochastic Gradient Descent و تعیین ضریب یادگیری مناسب بردارها را بهینه می‌کنند.

تناظر هر گره از گراف با یک بردار خود، کاری دشوار است اما آنچه پیاده سازی این دو الگوریتم را چالش برانگیز می‌کند سه موضوع است:

۱. تعداد مسیرهای تصادفی

۲. طول مسیرهای تصادفی

۳.طول بردار

طول بردار مهمترین بخش است چرا که در نمایش جزییات اطلاعاتی گراف نقشی تعیین کننده دارد. نکته حائز اهمیت بعدی تعداد مسیرهای تصادفی است. با توجه به بزرگ بودن شبکه، اجرای این دو الگوریتم به توان محاسباتی بسیار بالایی نیاز دارد و هر چند افزایش تعداد مسیرهای تصادفی معیارهای ارزیابی را افزایش می‌دهد اما محتاج توان محاسباتی به مراتب بیشتری است پس مطلوب است با وسواس تعیین گردد.

هر دو الگوریتم به صورت کتابخانه در زبان Python قابل استفاده هستند.برای استفاده کافی است آن‌ها را نصب کنید. در documentation هر دو الگوریتم نحوه نصب آن‌ها در سیستم عامل‌های مختلف ذکر شده است.

ورودی‌های هر دو الگوریتم شامل گراف یا لیست مجاورت یالی گراف، تعداد مسیرهای تصادفی، طول مسیرهای تصادفی و طول بردار است.

الگوریتم deepwalk از طریق دستورهای command در Anaconda Prompt اجرا می‌شود. کافی است یک فایل با پسوند adjlist. در path که شامل لیست مجاورت یالی گراف است و فایلی با پسوند embeddings. جهت دریافت بردارها قرار دهید. با اجرای command زیر الگوریتم اجرا خواهد شد.

```
deepwalk --input file.adjlist --number-walks 200 --representation-size
10 --walk-length 5 --window-size 10 –output file.embeddings
```

برای اجرای الگوریتم node2vec بر روی داده‌ها اما نیاز به کمی کد نویسی است. پس از import کتابخانه مربوطه در برنامه باید با ساختن objectای از کلاس node2vec ورودی‌ها را به آن داده و مدل را بسازید. باید مدل را در فایلی با پسوند emb. ذخیره نمایید.

تعاریف

برای اجرای دو الگوریتم از مجموعه داده Movie Lenz استفاده می‌کنیم. این مجموعه داده شامل ۱۰۰ هزار نمونه داده از میان ۹۴۳ کاربر و ۱۶۸۲ فیلم است. هر کاربر به فیلم‌هایی که تماشا کرده امتیازی از ۵-۱ داده است. هدف پیشنهاد فیلم‌هایی به هرکاربر با استفاده از دو الگوریتم ذکر شده و با در نظر گرفتن فیلم‌هایی است که تماشا کرده است. ابتدا فیلم‌هایی را انتخاب می‌کنیم که امتیاز بیشتر از ۴ از سوی کاربران دریافت کرده‌اند. به این ترتیب فیلم‌هایی در فاز یادگیری انتخاب می‌شوند که مورد توجه بیشتری از سوی هر کاربر بوده‌اند و در روند پیشنهاد بهتر به هر کاربر ما را یاری می‌کنند.

هر نمونه از این داده شامل سه بخش مختلف است. به طور مثال:

۲۴۳ ۱۱۸۴ ۲

این نمونه بیان می‌کند که کاربر شماره ۲۴۳ به فیلم ۱۱۸۴ امتیاز ۲ داده است. همانطور که اشاره شد اولین قدم در فاز یادگیری غربالگری فیلم‌های تماشا شده است که امتیاز بیشتر یا مساوی ۴ از سوی هر کاربر دریافت کرده است.

پیش از آن یک نکته قابل توجه است و آن اینکه کاربرها و فیلم‌ها هر دو از ۱ شماره گذاری شده‌اند و دو مجموعه شماره‌ها در بخشی هم پوشانی دارند. بنابراین، باید یک Mapping صورت گیرد به این شکل که فیلم‌ها از ۹۴۴ تا ۲۶۲۵ شماره‌گذاری گردند.

پس از تحقق امر بالا، یک ماتریس ۲۶۲۵*۲۶۲۵ در نظر می‌گیریم. با بررسی هم‌زمان مجموعه داده‌ها و ماتریس مفروض، چنانچه کاربری به فیلمی امتیاز بیشتر یا مساوی ۴ داده بود در خانه متناظر با سطر و ستون مربوطه ۱ و در غیر این صورت ۰ قرار می‌دهیم

واضح است که گراف شبکه مورد نظر یک گراف دو بخشی است و گره‌های یک بخش شامل کاربران و بخش دیگر شامل فیلم‌ها هستند. در اینجا دقت کنید که این گراف جهت‌دار نیست چرا که در صورت جهت‌دار بودن سر هر یال یک فیلم و دم آن یک کاربر را نشان می‌داد و برعکس. بدین‌گونه دوری میان دو بخش گراف موجود نبوده، برخی روابط از بین رفته و الگوریتم غیرقابل اجرا می‌بود.

حال ما ماتریس مجاورت این گراف دو بخشی را در اختیار داریم ولی ورودی هر دو الگوریتم لیست مجاورت یالی است. به آسانی می‌توان لیست مجاورت یالی را با استفاده از ماتریس مجاورت به دست آورد. به این صورت که در هر سطر از ماتریس مجاورت خانه‌هایی با مقادیر ۱ وجود دارند و بیانگر وجود یک یال بین دو سر گره‌ها هستند.

در قدم بعدی از فاز یادگیری برای هر کاربر داده آموزش و آزمون را با استفاده از ابزارهای کتابخانه Sklearn و نسبت ۸۰٪-۲۰٪ جدا می‌کنیم. این دو دسته هر یک بیانگر زیرگرافی القایی از گراف اصلی هستند و دقت کنید که باید ذخیره شوند چرا که هر دو الگوریتم باید چند مرتبه روی آن‌ها اجرا شوند.

با داشتن لیست مجاورت یالی زیرگراف داده آموزش می‌توان الگوریتم deepwalk را به ترتیب گفته شده اجرا نمود ولی برای اجرای الگوریتم node2vec برخلاف آن نیاز به خود زیر گراف داریم. در اینجاست که با استفاده از کتابخانه Networkx و با داشت لیست مجاورت یالی، زیر گراف را از روی آن ساخته و برای اجرا آماده می‌کنیم.

با اجرای دو الگوریتم بر روی داده آموزش فاز یادگیری پایان می‌یابد و بردارهای متناظر با هر گره از گراف را در اختیار داریم. در ادامه از ضرب کسینوسی و ضرب داخلی برای تعیین میزان شباهت بردارها استفاده می‌کنیم.

ضرب داخلی و ضرب کسینوسی دو بردار غیر صفر به ترتیب به طرق زیر تعریف می‌شوند:

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\| \cos\theta$$

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

هر قدر ضرب کسینوسی و ضرب داخلی دو بردار بیشتر باشد میزان شباهت آن‌ها بیشتر خواهد بود پس کافی است برای هر کاربر در شبکه، بردار متناظرش را در بردار فیلم‌هایی که در داده آموزش نبوده‌اند ضرب کنیم و مقادیرش را مرتب نماییم. به این ترتیب فیلم‌های پیشنهادی به ازای هر کاربر تعیین می‌گردند و وقت آن می‌رسد که Mapping را خنثی سازیم و برچسب هر فیلم را از بازه ۱۶۲۵-۹۴۴ به بازه ۱۶۸۲-۱ باز گردانیم.

فاز آزمون را با انتخاب n فیلم از فیلم‌های پیشنهادی به ازای هر کاربر آغاز می‌کنیم. برای هر کاربر آن‌ها را با داده آزمون که از قبل جدا کرده‌ایم مقایسه می‌کنیم.

معیارهای ارزیابی

Precision برای هر کاربر تقسیم اشتراکات داده آزمون و فیلم‌های پیشنهادی بر n تعریف می‌شود و برای مجموعه داده به صورت میانگین precision تمام کاربرها.

Recall برای هر کاربر تقسیم اشتراکات داده آزمون و فیلم‌های پیشنهادی بر تعداد داده‌های آزمون تعریف می‌شود و برای مجموعه داده به صورت میانگین Recall تمام کاربرها.

F1 برای هر کاربر به صورت ۲ برابر تقسیم حاصل ضرب بر حاصل جمع Precision و Recall آن کاربر تعریف می‌شود و برای مجموعه داده‌ها به صورت میانگین F1 تمام کاربرها.

NDCG برای هر کاربر به صورت تقسیم DCG بر IDCG و برای مجموعه داده ها به صورت میانگین NDCG تمام کاربرها تعریف می‌شود.

1-Call برای هر کاربر چنانچه تعداد اشتراکات داده آزمون و فیلم‌های پیشنهادی حداقل ۱ باشد، ۱ و در غیر این صورت ۰ تعریف می‌شود و برای مجموعه داده به صورت میانگین 1-Call تمام کاربرها.

با انتخاب پارامترهای زیر و معیار شباهت مورد نظر، هر الگوریتم را ۵ مرتبه اجرا می‌کنیم:

Number of walks = 200
length of walk = 5
dimension = 10
n = { 20, 50 , 100 }

deepwalk algorithm & inner product
number of walks = 200
length of walk = 5
dimension = 10
n = 20

	1	2	3	4	5	Mean
Precision	0.0026	0.003	0.0025	0.0027	0.003	0.00276
Recall	0.005	0.0071	0.006	0.006	0.0075	0.00632
F1	0.003	0.0038	0.0031	0.0032	0.0038	0.00338
NDCG	0.1525	0.1527	0.1523	0.1525	0.1525	0.1525
1-Call	0.0509	0.0593	0.0509	0.054	0.0583	0.05468

deepwalk algorithm & cosine similarity
number of walks = 200
length of walk = 5
dimension = 10
n = 20

	1	2	3	4	5	Mean
Precision	0.0266	0.0268	0.0273	0.027	0.0282	0.02718
Recall	0.0835	0.0861	0.0851	0.0855	0.0907	0.08618
F1	0.0364	0.037	0.0375	0.0371	0.0391	0.03742
NDCG	0.1549	0.1538	0.1539	0.1547	0.1546	0.15438
1-Call	0.3806	0.3775	0.3891	0.3806	0.3891	0.38338

node2vec algorithm & inner product
number of walks = 200
length of walk = 5
dimension = 10
n = 20

	1	2	3	4	5	Mean
Precision	0.0021	0.0019	0.0022	0.0023	0.0018	0.00206
Recall	0.0077	0.006	0.008	0.0077	0.0062	0.00712
F1	0.0031	0.0026	0.0032	0.0032	0.0025	0.00292
NDCG	0.1516	0.152	0.1518	0.1517	0.1519	0.1518
1-Call	0.0413	0.0381	0.0445	0.0445	0.0371	0.0411

node2vec algorithm & cosine similarity
number of walks = 200
length of walk = 5
dimension = 10
n = 20

	1	2	3	4	5	Mean
Precision	0.0009	0.0007	0.0007	0.0006	0.0007	0.00072
Recall	0.0032	0.0025	0.0026	0.0026	0.003	0.00278
F1	0.0013	0.0011	0.0011	0.001	0.0011	0.00112
NDCG	0.1517	0.1516	0.1516	0.1514	0.1514	0.15154
1-Call	0.019	0.0159	0.0159	0.0137	0.0148	0.01586

deepwalk algorithm & inner product
number of walks = 200
length of walk = 5
dimension = 10
n = 50

	1	2	3	4	5	Mean
Precision	0.0048	0.0047	0.0046	0.0048	0.0045	0.00468
Recall	0.0303	0.0317	0.0294	0.0312	0.295	0.08352
F1	0.0076	0.0075	0.0074	0.0077	0.0072	0.00748
NDCG	0.01	0.01	0.01	0.0099	0.0099	0.00996
1-Call	0.1919	0.1887	0.1866	0.1876	0.1834	0.18764

deepwalk algorithm & cosine similarity
number of walks = 200
length of walk = 5
dimension = 10
n = 50

	1	2	3	4	5	Mean
Precision	0.0271	0.0266	0.0277	0.0275	0.0281	0.0274
Recall	0.1903	0.1885	0.1933	0.1949	0.196	0.1926
F1	0.044	0.0432	0.045	0.0448	0.0455	0.0445
NDCG	0.0103	0.01	0.0102	0.0101	0.0102	0.01016
1-Call	0.6648	0.6405	0.6712	0.6765	0.6776	0.66612

node2vec algorithm & inner product
number of walks = 200
length of walk = 5
dimension = 10
n = 50

	1	2	3	4	5	Mean
Precision	0.0024	0.0023	0.0023	0.0023	0.0023	0.00232
Recall	0.019	0.0179	0.0182	0.0188	0.0194	0.01866
F1	0.0039	0.0038	0.0038	0.0039	0.0039	0.00386
NDCG	0.0098	0.0098	0.0097	0.0097	0.0097	0.00974
1-Call	0.1049	0.1039	0.1028	0.1028	0.1028	0.10344

node2vec algorithm & cosine similarity
number of walks = 200
length of walk = 5
dimension = 10
n = 20

	1	2	3	4	5	Mean
Precision	0.0008	0.0009	0.0009	0.0009	0.001	0.0009
Recall	0.0078	0.0087	0.0087	0.0085	0.0096	0.00866
F1	0.0014	0.0016	0.0016	0.0016	0.0017	0.00158
NDCG	0.0097	0.0098	0.0098	0.0097	0.0098	0.00976
1-Call	0.0424	0.0487	0.0455	0.0466	0.0466	0.04596

deepwalk algorithm & inner product
number of walks = 200
length of walk = 5
dimension = 10
n = 100

	1	2	3	4	5	Mean
Precision	0.0071	0.0069	0.0072	0.0073	0.0073	0.00716
Recall	0.0944	0.0945	0.0967	0.0994	0.0979	0.09658
F1	0.0125	0.0123	0.0127	0.013	0.0129	0.01268
NDCG	0	0	0	0	0	0
1-Call	0.422	0.4103	0.4167	0.422	0.0415	0.3425

deepwalk algorithm & cosine similarity
number of walks = 200
length of walk = 5
dimension = 10
n = 100

	1	2	3	4	5	Mean
Precision	0.0266	0.0258	0.0261	0.0269	0.0268	0.02644
Recall	0.3344	0.3244	0.3296	0.3346	0.3355	0.3317
F1	0.0468	0.0452	0.0459	0.0472	0.0471	0.04644
NDCG	0	0	0	0	0	0
1-Call	0.8472	0.8335	0.8398	0.8515	0.8494	0.84428

node2vec algorithm & inner product
number of walks = 200
length of walk = 5
dimension = 10
n = 100

	1	2	3	4	5	Mean
Precision	0.0021	0.0022	0.002	0.002	0.002	0.00206
Recall	0.0329	0.0341	0.0327	0.0295	0.03	0.03184
F1	0.0039	0.0039	0.0036	0.0036	0.0037	0.00374
NDCG	0	0	0	0	0	0
1-Call	0.176	0.1749	0.1643	0.158	0.1643	0.1675

node2vec algorithm & cosine similarity
number of walks = 200
length of walk = 5
dimension = 10
n = 100

	1	2	3	4	5	Mean
Precision	0.001	0.001	0.001	0.0009	0.0008	0.00094
Recall	0.0154	0.0172	0.0168	0.0152	0.0113	0.01518
F1	0.0018	0.0019	0.0018	0.0017	0.0014	0.00172
NDCG	0	0	0	0	0	0
1-Call	0.0922	0.0943	0.0954	0.0848	0.0721	0.08776

مراجع

- Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, *151*, 78-94.
- Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, August). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 701-710).
- Grover, A., & Leskovec, J. (2016, August). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 855-864).

مراجع

- Cai, W., Pan, W., Liu, J., Chen, Z., & Ming, Z. (2020). k-Reciprocal nearest neighbors algorithm for one-class collaborative filtering. *Neurocomputing*, *381*, 207-216.

با سپاس از توجه شما

شکیبا طایفه