دانشگاه صنعتي اميركبير

(پلی‌تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

درس مباحثی در ریاضیات و کاربردها:

کاربرد جبرخطی در علم داده

Transforming PageRank Into An Infinite-Depth Graph Neural Network

نگارش

گروه ۱۰

استاد

دکتر فاطمه شاکری

خرداد ۱۴۰۴

چکیده

با وجود موفقیت های معماری های یادگیری عمیق [1]در حوزه های مختلف، همچنان مدل های یادگیری گراف[2]، مدل هایی کم عمق هستند. این موضوع باعث کاهش ظرفیت و عدم توانایی در یادگیری و دریافت روابط دور برد در این مدل از شبکه ها می‌شود. علت اصلی طراحی کم عمق این شبکه ها، بیش‌هموارسازی[3] است بوده که در صورت افزایش عمق شبکه باعث یکسان شدن گره های گراف می‌شود.

در این پژوهش قصد داریم ایده ی مقاله ای به نام 'تبدیل پیج رنک به شبکه یادگیری گراف با عمق نامتناهی' [4]را بررسی و پیاده سازی کنیم. در این مقاله ذکر شده است که ایده اصلی براساس شباهت نزدیک ایده های شبکه های یادگیری گراف با رتبه‌صفحه [5]بنا شده است. یک انشعاب از ایده اصلی رتبه‌صفحه ، رتبه‌صفحه شخصی سازی شده [6]است که جدای از ایده اصلی، یک بردار شخصی درنظر گرفته می‌شود. با درنظر گرفتن این انشعاب، ایده ی شبکه عصبی رتبه‌صفحه گراف شخصی سازی شده معرفی شده است که یک توسعه از شبکه های پیچش گراف [7]به یک مدل با عمق نامتناهی که به ازای هر جمع آوری اطلاعات همسایگان گره ها، شانس بازگشت به حالت اولیه در هر گردش وجود خواهد داشت. در این مدل اثبات می‌شود که تا عمق نامحدود، مدل در نهایت به جواب یکتا میل خواهد کرد و به عنوان نتیجه مشکل بیش‌هموارسازی حل خواهد شد. همچنین پیچیدگی زمانی همچنان خطی خواهند ماند و با ارائه های روش های بهینه سازی در محاسبه گرادیان ها، پیچیدگی حافظه ثابت و مستقل از عمق شبکه خواهد شد. این مدل معماری

[1] Deep learning

[2] Graph Neural Networks

[3] Over-smoothing

[4] Transforming PageRank Into An Infinite-Depth Graph Neural Network

[5] PageRank

[6] Personalized PageRank

[7] Graph Convolutional Network

مستعد استفاده در گراف های بسیار بزرگتر خواهد بود. در نهایت بهینگی این معماری در وظیفه های مختلف طبقه بندی گره ها و گراف ها نشان داده خواهد شد.

واژه‌های کلیدی:

یادگیری ماشین، شبکه های عصبی گراف، رتبه‌صفحه

فهرست مطالب

صفحه

۱. فصل اول

مقدمه

ساختارهای داده گراف در بسیاری از حوزه های دنیای اطراف ما همانند شبکه های اجتماعی یا ساختار های زیستی قابل مشاهده هستند. رشد حجم داده های اینگونه باعث افزایش تقاضا برای مدل هایی شده که توانایی استخراج اطلاعات غنی فراوانی از این داده ها دارند. وظیفه هایی مانند سیستم های پیشنهاد کننده، پیشبینی وضعیت ترافیک، یا طبقه بندی کل یک گراف در دسته بندی های مختلف، همگی از معدود وظیفه هایی هستند که توجه محققان را به خود جلب کرده است. رویکرد هایی مبنی بر یادگیری عمیق، موفقیت های زیادی در داده های مشبک [1]دست یافته اند که برای مثال میتوان به پردازش تصویر و پردازش زبان طبیعی اشاره کرد. شبکه های عصبی گراف ایده های پیچش در فضای اقلیدسی برای دامنه های غیر معمول غیر اقلیدسی را به کار گرفته اند. این روش ها به طور مستقیم ساختار گراف را حین پیچش درنظر میگیرند.

یکی از چالش های GNN ناتوانی در دریافت و درک وابستگی های دور برد است. روش های محبوبی که به تازگی معرفی شده اند، از یک رویکرد جمع آوری[2] اطلاعات که در آن از k لایه پیچش گراف که منجر به جمع آوری اطلاعات همسایگان با فاصله ی k گام از هر گره میشود، استفاده میکنند. چندین مشکل اصلی در این روش وجود دارد که مهمترین آنها مشکل بیش‌هموارسازی و مصرف حافظه است.

چندین راه حل برای این مشکلات در مقاله های مرتبط معرفی شده است ولی یا دارای محدودیت های فراوانی بر ورودی ها هستند یا عمق نامتناهی را نمیتوانند هندل کنند.

در مقاله ی (Günnemann, 2018 & ,Klicpera, Bojchevski) ارتباط بسیار نزدیک GNN ها و رتبه‌صفحه بررسی شده است. نسخه اولیه رتبه‌صفحه که توسط (Page, 1999) معرفی شده است، تنها بر ساختار کلی گراف تمرکز کرده و هیچ تاثیری از اطلاعات هر گره نمیگیرد. نسخه شخصی سازی شده رتبه‌صفحه (Page, 1999) شانس بازگشت به یک بردار تلیورت اولیه را

[1] Grid-Structured data

[2] Aggregation

معرفی میکند که منجر میشود نتایج نه تنها به ساختار گراف وابسته باشد بلکه بر توزیع ابتدایی داده شده نیز تکیه کند.

در راستای اثبات میل کردن شبکه معرفی شده PPRGNN به یک پاسخ یکتا در عمق نامتناهی، احتمال ریست شدن به توزیع ابتدایی در هر جمع آوری اطلاعات همسایگی هر گره را، به صورت پویا و وابسته به عمق لایه تنظیم میکنیم. بدین صورت که با افزایش عمق شبکه، شانس ریست نشدن کاهش خواهد یافت. حائز اهمیت است که شبکه معرفی شده دیگر از مشکل بیش‌هموارسازی یا تاثیر صرف گرفتن از اطلاعات همسایگی محلی، رنج نمیبرد. اما باید ذکر کنیم که به دلیل عمیق بودن لایه ها و عمق ها، همچنان اطلاعات همسایگان دور تر نتایج نمایش ویژگی های هر گره را تحت تاثیر قرار میدهد.

همچنین با اینکه از نظر تئوری عمق نامتناهی این معماری اثبات میشود ولی در عمل، عمق بهینه وابسته به ساختار گراف، پارامتر های یادگرفته شده و ویژگی های مشاهده شده می‌باشد. همچنین ما یک راه برای کنترل کردن نرخ میل کردن شبکه معرفی میکنیم، چرا که برای انواع مختلف ساختار های گرافی، عمق های بهینه و میزان محلی بودن خاصی موجود است.

شایان ذکر است که مدل ما برخلاف مدل های قبلی با عمق نامتناهی، نیازی به هیچ شرطی محدود نیست.

برای فراهم کردن امکان مقیاس دهی شبکه به گراف های بسیار بزرگ، ما یک سیستم بهینه برای محاسبه گرادیان طراحی کرده ایم که هزینه زمانی و حافظه ای را به صورت ضریبی ثابت حفظ میکند.

برای نشان دادن توانایی رویکرد ارائه شده، دو وظیفه inductive و transductive را با مدل های دیگر موجود در این حوزه اجرا کرده و نتایج را گزارش خواهیم کرد. مدل ما تقریبا تمامی وظیفه ها را با اختلافی قابل توجه عملکرد بهتری از خود نشان می‌دهد. همچنین زمان اجرا نیز نسبت به مدل های عمق نامتناهی دیگر بهبود یافته است.

در بخش دوم قرارداد ها و تعارف، مفاهیم اولیه و مربوطه به رتبه‌صفحه شخصی سازی شده و شبکه های عصبی گراف را بررسی می‌کنیم، در بخش سوم نیز رویکرد های مشابه و مربوطه را بررسی می‌کنیم و در بخش چهارم رویکرد ما با جزئیات مناسب بررسی شده و در بخش آخر نتایج گزارش خواهد شد.

۲. فصل دوم

Preliminaries

ما یک گراف را به شکل $G = (V, E)$ که یک چندتایی از n گره به شکل $V = \{v_1, v_2, \dots, v_n\}$ و یک مجموعه از یال ها که با E نمایش داده می‌شود و مجموعه ای از دوتایی هایی است که از دو گره تشکیل شده اند. یک ماتریس مجاورت را که به شکل $A \in R^{n \times n}$ نمایش داده می‌شود، اتصال هر دو گره در E را نشان می‌دهد. ورودی $a_{ij} \in A$ قدرت یک یال بین رئوس v_i و v_j را نشان می‌دهد،یک ورودی صفر نشان دهنده عدم وجود یک یال است. ما از نسخه نرمالایز شده $\tilde{A} = D^{-1/2} A D^{1/2}$ ماتریس مجاورت استفاده می‌کنیم که احتمالا دارای طوقه است. هر گره مثل v_k دارای یک مجموعه از ویژگی های F بوده که به صورت $u_k \in R^F$ نمایش داده می‌شود. ماتریس ویژگی ها که به صورت $U \in R^{n \times F}$ نمایش داده می‌شود شامل تمام ویژگی های گره ها است که به صورت پشته ای از تمام u_k ساخته شده است.

مجموعه تمام همسایگی هر گره را به صورت $N_i = \{v_j | \tilde{A}_{ij} > 0\}$ محاسبه می‌شود، مجموعه تمام گره های متصل به v_i را نشان میدهد.

۱-۲- رتبه‌صفحه شخصی سازی شده[1]

ما ایده اصلی خود را از رتبه‌صفحه شخصی سازی شده که توسط (Page, 1999) ارائه شده است، به طور مختصر در این بخش مرور خواهیم کرد. الگوریتم رتبه‌صفحه (Page, 1999) در ابتدا برای امتیاز دهی و تشخیص میزان اهمیت صفحات موجود در وب طراحی شد. هر صفحه موجود در وب نمایان گر یک گره در گراف اینترنت بوده و لینک های بین آنها به عنوان یال های جهت دار درنظر گرفته میشود. نقطه همگرایی معادله زیر پاسخ الگوریتم رتبه‌صفحه است:

$$r = Ar$$

[1] Personalized PageRank

که $r \in R^n$ بردار ویژه غالب ماتریس A است. بردار r میتواند با استفاده از روش توانی با بردار اولیه r_0 حاصل شود، اما قبل از آن چندین مرحله در جهت پایداری و یکتایی بردار حاصل شده باید صورت بگیرد که جزئیات آن خارج از حوصله این مقاله است.

برای تفسیر بهتر این روش، می‌توانیم ماتریس A را به عنوان یک ماتریس انتقال تصادفی به روی گراف داده شده درنظر بگیریم که به ما امکان انجام یک گشت تصادفی [1]را می‌دهد.

در نتیجه توزیع احتمالاتی ایستای حاصل شده توسط گشت تصادفی همان بردار r در حد بینهایت خواهد بود. یکی از نتایج این ارتباط این است که بردار r تنها بر ساختار کلی گراف وابسته بوده و به هیچ اطلاعات قبلی موجود از گره ها اهمیتی نمی‌دهد. در نتیجه نویسندگان مقاله رتبه‌صفحه، نسخه شخصی سازی شده این الگوریتم را نیز ارائه کرده اند:

$$r = (1 - \alpha)Ar + \alpha u$$

که به معادله اولیه، شانس بازگشت و پرش به بردار شخصی سازی شده ای که نمایانگر توزیع احتمالاتی فراهم شده است، را فراهم می‌کند. بردار شخصی سازی $u \in R^n$ توزیع ابتدایی [2]صفحات وب را نشان میدهد. تفسیر مرتبط با گشت تصادفی این معادله بدین صورت است که هر گشت تصادفی شانس بازگشت به وضعیت بردار شخصی سازی را خواهد داشت.

۲-۲- شبکه عصبی گراف[3]

مفهوم دیگری که از آن استفاده کرده ایم، شبکه های عصبی گراف ها (GNN) و به طور خاص زیرشاخه ای از آن به نام شبکه های عصبی ارسال پیام[4] (MPNNs) (Justin, Schoenholz, Riley, Vinyals, & Dahl, 2017) می‌باشد. GNN ها از عملیات های Permutation equivariant به روی ساختار داده های گراف استفاده می‌کنند. با الهام گیری از پیچش های طیفی گراف ها به عنوان یک تقریب رتبه اول محلی، عملیات انتقال پیام نیز، وضعیت هر گره مثل h_i

[1] Random Walk

[2] Initial Distribution

[3] Graph Neural Networks

[4] Message-Passing Neural Networks

را با ترکیب اطلاعات گره های همسایگی مستقیم آن گره به روزرسانی میکند. تابع به روزرسانی را می‌توان به شکل زیر نمایش داد:

$$h_i^{(l+1)} = \psi\left(h_i^{(l)}, \sum_{j \in N_i} \omega(h_i^{(l)}, h_j^{(l)})\right)$$

دقت کنید که فرمول بالا، تابع آپدیت را به صورت گره محور نمایش میدهد، همچنین برای وضعیت نهان هر گره $h_i^{(l)}$، با استفاده از توابع آموزش پذیر ω و ψ به همراه یک تابع جمع آوری permution invariant که با $\bigoplus$ یا summation نمایش می‌دهیم. در این مقاله از یک نسخه بسیار اولیه از شبکه عصبی یادگیری گراف به اسم شبکه پیچشی گراف یا به اختصار GCN استفاده میکنیم. (Kipf & Welling, 2016)

پیش از ادامه خوب است کمی درباره ی Permutation invariant صحبت کنیم:

تصور کنید که یک شبکه عصبی MLP در اختیار داریم، یک راه اولیه برای اجرای مدل های یادگیری عمیق به روی این گراف ها، این است که ماتریس مجاورت را به شکل بردار سطری یا ستونی در بیاوریم و سپس آن را به شبکه عصبی پرسپترون چند لایه بدهیم، شفاف است که هر جایگشت متفاوتی از این درایه های این بردار، یک نتیجه جدید در شبکه حاصل خواهد کرد یا به اصطلاح این مدل Permutation invariant نیست.

به صورت ریاضی نیز میتوان permutation invariant را به شکل زیر نشان دهیم:

هر تابعی مثل f که با دریافت ماتریس مجاورت A و ماتریس جایگشت P به شکل زیر باشد را یک تابع permutation invariant می‌نامیم:

$$f(PAP^T) = f(A)$$

و permutation equivariance را به صورت ریاضیاتی به شکل زیر تعریف می‌کنیم:

$$f(PAP^T) = Pf(A)$$

مدل های یادگیری عمیق مناسب برای گراف ها حتما باید یکی از دو خواص Permutation invariant یا Permutation equivariance را دارا باشد.

حال که با نمایش تابع به روزرسانی گره محور یک مدل پایه ای یادگیری گراف مثل GCN آشنا شدیم، میتوانیم اکنون به نمایش ماتریسی آن بپردازیم:

$$H^{(l+1)} = \Phi\left(\tilde{A}H^{(l)}W^{(l)}\right) (4)$$

که $W \in R^{d\times d}$ یک تبدیل خطی، Φ یک تابع فعالساز عضو محور بوده و $H^{(l)} \in R^{n\times d}$ شامل وضعیت گره ها $(h_i^{(l)})$ پس از لایه l ام است.

همچنین حائز اهمیت است که هر لایه تنها اطلاعات همسایه های یک گره را جمع آوری می‌کند، در نتیجه بعد از k لایه، تنها اطلاعات همسایه های k فاصله، قابل دسترس است. با در دسترس بودن این موضوع، به آسانی نتیجه گیری می‌شود که با انتخاب k دلخواه، اطلاعات لایه $k+1$ در دسترس نخواهد بود. اما لازم است بدانیم که ما قادر نیستیم اندازه ی k تا هر حد دلخواهی افزایش دهیم، چرا که مشکلات متعددی پیش خواهد آمد که در دو زیربخش بعدی به آن می‌پردازیم.

۱-۲-۲- بیش‌هموارسازی

یافته های تحقیقاتی اخیر نشان می‌دهد که با انباشت لایه های جمع آوری اطلاعات در مدل های یادگیری گراف، منجر به کاهش عملکردشان شده که علت آن پدیده ای به اسم بیش‌هموار سازی است.

محققان نشان داده اند که معادله (۴) حالتی خاصی از هموارسازی لاپلاسی است که منجر می‌شود نمایش گره ها با افزودن لایه های بیشتر، به یکدیگر شبیه تر شوند. همچنین اثبات می‌شود که هموارسازی لاپلاسی به ترکیب خطی ای از بردار های ویژه ی غالب همگرا خواهد شد. با اینکه مقدار کمی هموارسازی برای پخش کردن اطلاعات بین گره ها مورد نیاز است، با این حال نمایش گره ها با هموارسازی بیش از حد، از یکدیگر قابل تمایز نخواهند بود، در نتیجه منجر می‌شود دقت پیشبینی های داده محور کاهش یابد.

همچنین ارتباط نزدیکی بین k لایه از معادله (۴) به یک گشت دلخواه به طول k وجود دارد و این مهم بدین معنی است که هر دو معادله در نهایت به توزیع حدی ای از گشت دلخواه همگرا خواهند شد و در نتیجه این گشت دلخواه از گره ریشه و شروع کننده مستقل خواهد شد و به اطلاعات محلی گره ها اعتنایی نخواهد کرد و در نهایت اطلاعات مخصوص به هر گره تاثیری در وضعیت نهایی نخواهد داشت که این موضوع منجر به کاهش عملکرد مدل ها با افزایش لایه ها و طول

گشت دلخواه خواهد شد. در اصل در بسیاری از اجرا ها، افزایش تعداد لایه از ۲ به بعد، منجر به کاهش عملکرد می‌شود.

۲-۲-۲- پیچیدگی حافظه

مشـــکل دیگری که جلوی GNN ها را از یک مدل عمیق بودن می‌گیرد، پیچیدگی حافظه اســـت. گراف ها به راحتی میتوانند بیش از میلیون ها گره داشته باشند، که به دلیل پیچیدگی حافظه ای از مرتبه $O(kn)$ که k تعداد لایه ها و n تعداد گره ها است، منابع حافظه موجود به راحتی مصرف می‌شوند.

چندین راه حل براساس نمونه گیری همسایه های محلی هر گره در جهت بهبود مرتبه حافظه مدل ها معرفی شده اند. بر اساس پدیده ای به اسم انفجار همسایگی[1]، تعداد گره های k-فاصله از هر گره دلخواه از مرتبه $O(d^k)$ خواهد بود که d میانگین درجه هر گره در گراف مدنظر اســـت. درنتیجه برای تعداد لایه های زیاد k، این روش نیز ناکارآمد خواهد بود.

روش دیگر براساس خوشه بندی گراف به چندین زیرگراف و استفاده از آنها برای آموزش شکل گرفته است. حائز اهمیت است که این روش نیز نمیتواند از پتانسیل کامل یک گراف استفاده کند و بخشی از اطلاعات و پیوند های موجود بین زیرگراف ها از بین خواهد رفت.

در نتیجه این مشکلی است که در طراحی هر مدل یادگیری گراف نیازمند توجه است.

[1] Neighborhood Explosion

۳. فصل سوم
تحقیقات مرتبط پیشین

تحقیقات زیادی در زمینه افزایش عمق مدل های MPNN با توجه به مشکل بیش‌هموارسازی و پیچیدگی حافظه پرداختند. به عنوان مثال (Rong, Huang, Xu, & Huang, 2020) متوجه شدند که پدیده بیش‌هموارسازی برای گره هایی که یال های ورودی بیشتری دارند زودتر اتفاق خواهد افتاد و برای مقابله با آن، DropEdge را معرفی کردند که مشابه ایده ی dropout در شبکه های عصبی معمول هستند. آنها به طور تصادفی یال ها را نمونه گیری کرده و سپس آن هارا هنگام ایپاک های آموزشی رها حذف می‌کنند و نشان میدهند که این ایده برای وظیفه های نیمه-با ناظر طبقه بندی گره ها برای گراف هایی با هموفیلی بالا عملکرد خوبی داشته ولی برای گراف ها پیچیده تر به مشکل میخورد.

با الهام گیری از ایده ی ResNet که توسط (He, Zhang, Ren, & Sun, 2016) معرفی شده، بعد ها شبکه ی GCNII معرفی شد که از ارتباطات باقی مانده از دو طریق استفاده می‌کند. در هر لایه آنها یک ارتباط اولیه باقی مانده با وضعیت اولیه $H^{(0)}$ و یک ارتباط همانی با وزن های آن لایه اضافه کرده اند و نشان داده اند که این ایده باعث بهبود عملکرد می‌شود.

با اینکه تمامی این روش ها اثر بیش‌هموار سازی را کاهش میدهند ولی همچنان با یافتن عمق موثر مشکل دارند.

۱-۳- شبکه های عصبی گراف با عمق نامحدود[1] (IDGNNs)

چندین آپشن در رویکرد های متفاوت توسط محققان در زمینه اجرای پیچش در شبکه های عصبی گراف با عمق نامحدود بررسی شده اند. همه ی این روش ها از اعمال فیلتر های گراف و پیچش تا زمانی که همگرایی به پاسخ رخ دهد استفاده می‌کنند. زمانی که از معادله یک شبکه عصبی گراف با عمق نامحدود استفاده میکنیم نیاز داریم تا پاسخ در نهایت به جوابی یکتا همگرا شود. ما این همگرایی را خوش وضعی تعریف کرده و گوییم:

(تعریف ۳.۱) خوش وضعی: با در دست داشتن ماتریس $X \in R^{N\times D}$ به عنوان ورودی، معادله $Y = g(X)$ با تابع g که یک تابع بازگشتی با عمق نامتناهی است را خوش وضع گوییم هرگاه:

[1] Infinite-Depth Graph Neural Networks

۱ – پاسخ Y یکتا باشد.

۲ – $g(X)$ به پاسخ یکتای Y همگرا شود.

درحالی که خوش وضعی معادله به روزرسانی GCN به صورت کلی تضمین نمی‌شود، پژوهش ما یک معادله مشابه معرفی خواهد کرد که خوش وضع بودن آن اثبات میشود. درجهت دستیابی به اهداف خویش ابتدا دو تحقیق مهم که نقطه آغاز ایده ی ما است را بررسی میکنیم.

APPNP -۳-۲-۱

(Günnemann, 2018 & ,Klicpera, Bojchevski) یک روش propagation که از ایده ای رتبه‌صفحه شخصی شده الهام گرفته شده، ارائه کرده اند. آنها ارتباط بین توزیع حدی MPNN ها و رتبه‌صفحه را شناسایی کرده و نشان دادند که هر دو رویکرد در نهایت تمرکز خود را از توزیع اولیه وضعیت گره ها از دست میدهند. همانطور که رتبه‌صفحه شخصی سازی شده به عنوان راه حل این مشکل برای الگوریتم اولیه رتبه‌صفحه معرفی شده بود، آنها همین ایده را به روی MPNN ها بکار بردند. آنها بردار شخصی سازی را وضعیت اولیه همه ی گره ها در نظر گرفتند، سپس شانس پرش و تلپورت به وضعیت اولیه را توسط یک پارامتر قابل بهینه سازی را به معادله ی یادگیری شبکه عصبی گراف اضافه کردند. در نهایت با سرهم کردن این دو ایده، مدل Approximate Personalized Propagation of Neural Predictions معرفی شد.

معادله انتقال پیام ها به شکل زیر است:

$$H^{(l+1)} = (1 - \alpha)\tilde{A}H^{(l)} + \alpha H^{(0)}$$

این معادله به طور تکراری و احتمالا به صورت نامحدود میتواند اطلاعات همسایگی را جمع آوری کند. همچنین آنها شانس بازگشت به وضعیت اولیه همه ی گره ها را با $H^{(0)} = f_\theta(U)$ که خروجی تابع f، وضعیت خروجی لایه قبل است، به معادله اضافه کرده اند. آنها همچنین نشان دادند که این معادله برای هر α بین ۰ و ۱ و ماتریس های $H^{(0)} \in R^{N\times D}$ و $\tilde{A} \in R^{N\times N}$ به شرطی که $\det(\tilde{A}) \leq 1$ باشد، خوش وضع است. دقت کنید که نرمال سازی ماتریس مجاورت نقش مهمی در خوش وضعی دارد. همانطور که قابل مشاهده است معادله ارائه شده شامل هیچ

پارامتر قابل یادگیری نبوده و آنها در نسخه ی دیگری معادله ی مشابهی ارائه کرده اند که تابع f اعمال یک شبکه عصبی پرسپترون چند لایه به روی گره های گراف باشد. این روش تنها برای وظیفه نیمه با ناظر طبقه بندی گره های گراف ارائه شده است.

در نهایت با اعمال یک تابع فعالساز *softmax* ، خروجی آخرین گردش معادله را به پیشبینی طبقه بندی تبدیل میکند.

۲-۱-۳- شبکه های عصبی ضمنی گراف[1]

در این مدل که از ایده ی کلی ضمنی سازی پیچش های گراف بوجود آمده است، از یک روش نقطه ثابت برای معادله غیر خطی زیر برای اثبات خوش وضع بودن استفاده میکنند:

$$X = \Phi\left(WX\tilde{A} + f_\theta(U)\right)$$

درحالی که در حالت کلی خوش وضعی این معادله تضمین نمیشود، محققان خوش وضعی این معادله را برای حالت خاصی که از وضعیت های ماتریس A و W را استفاده میکند، ثابت کرده اند. باید توجه کنیم که محدودیت هایی که برای این دو ماتریس درنظر گرفته اند، این مدل را برای تعمیم ناکارآمد میکند.

[1] Implicit Graph Neural Networks

۴. فصل چهارم
شبکه عصبی رتبه‌صفحه گراف

همانطور که در فصـول گذشـته به کرات توضـیح دادیم، پاسـخ الگوریتم رتبه‌صـفحه، یک توزیع احتمالی حدی اسـت که فارغ از توزیع گره ها اسـت. همچنین گفتیم که به دلیل ارتباط بین MPNN و رتبه‌صـفحه، شـبکه های عصـبی انتقال پیام نیز تاثیری از اطلاعات محلی و ویژگی های گره ها نمیگیرند.

الگوریتم رتبه‌صـفحه شـخصـی سـازی شـده نیز در جهت مقابله با از بین رفتن تمرکز بر اطلاعات محلی گره ها معرفی شد.

ما با الهام گیری از تحقیقات گذشـته، شـبکه عصـبی رتبه‌صـفحه شـخصـی سـازی شـده گراف ها (PPRGNN) را معرفی میکنیم. این مدل نیز با معرفی وضـعیت ابتدایی گره ها و شـانس بازگشت به آن در هر گردش، حفظ اطلاعات محلی گره ها را تضـمین میکند. همچنین همگرایی این مدل به یک پاسـخ یکتا نیز اثبات می‌شـود. درنتیجه مدل ما میتواند عمق نامحدودی بدون اینکه از مشـکل بیش‌هموارسـازی رنج ببرد را کاوش کند. در عمل نیز ما آنقدر پیچش به روی گراف انجام میدهیم تا دیگر تغییر محسـوسـی در پاسـخ نهایی صـورت نگیرد و پاسـخ به توزیع حدی گره ها نزدیک باشد. در این تحقیق ما GCN را به عنوان پایه سازنده ایده خود انتخاب میکنیم و منطقی است که این ایده قابل تعمیم به تمام زیر شاخه های شبکه های عصبی گراف باشد.

ما شـانس حرکت و کشـف گسـترده تر گراف را با α_l نمایش میدهیم. فرمول به روزرسـانی یک شـبکه GCN را با الهام از رتبه‌صـفحه شـخصـی سـازی شـده، بازنویسـی می‌کنیم و به رابطه زیر میرسیم:

$$H^{(l+1)} = \Phi(\alpha_l \tilde{A} H^{(l)} W + f_\theta(U)) \quad (7)$$

با حالت اولیه $H^{(0)} = 0$ که ماتریس قابل یادگیری W را بدون هیچ قید و شـــرطی به اشـــتراک میگذارد.

دقت کنید که به دلیل خاصـیت بازگشـتی رابطه ارائه شـده و عدم قید ماتریس W به رشـد نمایی و انفجاری این ماتریس خواهیم رسـید که به ازای α_l ثابت، خوش وضـع بودن مدل را به هم میزند. مشکل عدم ضمانت همگرایی بدین دلیل است که گره های دور تر از گره مورد بررسی، در توان

های بزرگتری از W ضـرب شـده که باعث غالب شـدن این گره ها در پاسـخ نهایی می‌شـود. همانطور که در الگوریتم رتبه‌صفحه نیز دیدیم، توزیع حدی نیز در نهایت به ساختار گراف وابسته بوده و به اطلاعات محلی و ویژگی های گره ها توجهی ندارد که منجر به از دست رفتن اطلاعات مهم و ناقص بودن پاسخ نهایی می‌شود.

ایده اصلی ما برای ضمانت همگرایی رابطه ارائه شده بدون محدود کردن پارامتر ها این است که احتمال ادامه دادن کشـف و بررسـی گراف به ازای افزایش عمق، کاهش یابد. همانطور که انتقال پیام در شبکه های عصبی گراف به یک گشت دلخواه مرتبط است، میتوان اینگونه تفسیر کرد که به ازای افزایش طول گام مسیر رندوم، شانس ریست شدن گشت دلخواهمان نیز افزایش یابد. پس از بررسـی ها نیز دریافتیم که اسـتفاده کردن از یک شـانس کاهشـی مثل $\alpha_n = \frac{1}{n}$ برای ضـمانت همگرایی مدل به یک راه حل یکتا کافی اسـت. ذات بازگشـتی رابطه ارائه شـده منجر به ضـرب تمامی α_n شـده که باعث میشـود اثر کاهندگی مدل در حد به $\frac{1}{n!}$ میل کند. پس با رشـد نمایی ماتریس W که در حد به W^n میل میکند، اثر کاهندگی شانس تلپورت به حالت اولیه رشد بیشتری داشـته و در نتیجه همگرایی رابطه اثبات میشـود. در جهت اسـتفاده موثر تر از عمق و پایداری عددی و کنترل سرعت همگرایی، شانس تلپورت را به صورت $\frac{1}{1+n\varepsilon}$ معرفی میکنیم و به صورت رسمی اثبات میشود که همگرایی برای هر$\varepsilon > 0$ درنهایت رخ خواهد داد.

دقت کنید که شانس بازگشت به وضعیت اولیه گره ها را عدد ثابت ۱ درنظر میگیریم (مستقل از افزایش عمق) چرا که در حد شـانس تلپورت $(1 - \alpha_l)$ برای همسـایه های نزدیک بسـیار کوچک خواهد شـد. که منجر به همان مشـکل بیش‌هموارسـازی که در بخش ۲.۲.۱ توضـیح دادین خواهد شد.

با قرار دادن شـانس تلپورت در معادله (۷) کشـف کردیم که دورترین گره ها همان ابتدا پردازش خواهند شد و گره هایی که در همسایگی مستقیم گره شروع قرار دارند در مرحله ی آخر پردازش میشوند. در نتیجه با اعمال ماتریس مجاورت نرمال شده که به صورت نمایی در خود ضرب شده، منجر به تغییر بسیار شدید وضعیت اولیه گره ها خواهد شد. در نتیجه برای محاسبه ی $H^{(1)}$، نیاز داریم که α_0 مینیمال باشد که با سـازوکار کنونی α_l است، چرا که برای رسیدن به مینیمال بودن α_l باید ارتباط معکوس n و l را هندل کنیم.

با فرض درنظر گرفتن تعداد کل ایتریشـــن ها که با k نمایش میدهیم، میتوانیم رابطه α_l به شـــکل $\frac{1}{1+(k-l-1)\varepsilon}$ برای هر لایه l تعریف کنیم که مشکل مطرح شده در پاراگراف قبل را هندل میکند. با اینکه میتوانیم با همین معرفی مستقیما به سراغ پیاده سازی و اثبات های ایده برویم ولی لازم است یادآوری کنیم، حالتی که برای ما حائز اهمیت است زمانی است که $k \to \infty$. چرا که ایده ی اصلی در راستای افزایش عمق شبکه های عصبی گراف به صورت نامحدود است.

در نهایت برای بررســـی تئوریکال همگرایی معادله ۷ به طوری که عمق ناحدود و فارغ از k را اختیار کنیم، نیاز اســت که اندیس گذاری کلی متغیر ها را به صــورت $n = k - l$ تغییر دهیم که منجر میشــود به معادله ی زیر برســیم (دقت کنید که این معادله حالت *flipped* شــده ی معادله ۷ است):

$$G^{(n)} = \Phi(\beta_n \tilde{A} G^{(n+1)} W + f_\theta(U)) \quad (8)$$

که $\beta_n = \alpha_{k-l-1}$ و $G^{(0)} = H^{(k)}$ بـه ازای هر k دلخواهی کـه در هر دو G و H بـه طور یکسان به کار رفته است. محاسبه ی $G^{(n)}$ را میتوان با استفاده از $G^{(n+1)}$ بدون اطلاع از k آسان اسـت. همین موضـوع باعث میشـود که گسـترش و پردازش این معادله بازگشـتی با عمق نامحدود بدون نیاز به تعیین مقدار ثابت k شدنی باشد.

برای آسـانی نگارشـمان قرار داد میکنیم که $G^{(l:k)}$ به ازای $G^{k+l+1} = 0$ معادل است با اجرای k ایتریشن برای محاسبه ی $G^{(l)}$.

قضـیه ۱: پاسـخ $G^{(l:k)}$ که از معادله$G^{(n)} = \Phi(\beta_n \tilde{A} G^{(n+1)} W + B)$ و پارامتر $\beta_n = \frac{1}{1+n\varepsilon}$ به ازای هر $l \in R$، $W \in R^{d \times d}$ ، $\tilde{A} \in R^{n \times n}$ ، $B \in R^{d \times n}$ ، $n \in N$ ، هر ε مثبت و هر تابع فعالساز پیوسته لیپشیتز Φ، زمانی که k به بینهایت میل کند، به یک پاسخ یکتا همگرا می‌شود.

بـه طور کـاربردی هر دو معادلـه (۷) و (۸) از دورترین گره ها شـــروع میکنند، کـه برای این منظور مقدار دقیق k مورد نیاز اسـت. این یک چالش اسـت، چرا که ما اطلاعی از برقرار شـدن شــرط پایان و همگرایی نداریم. از آنجایی که وضــعیت حدی $G^{(0:k)}$ زمانی که k به بینهایت میل میکند برای ما حائز اهمیت اســـت، میتوانیم با درنظر گرفتن یک مقدار مرزی مثل ε اقدام به شناسایی تعداد ایتریشن های کافی با رابطه زیر کنیم:

$$k = \min\{M \mid G^{(0:k)} - G^{(0:k+1)} < \varepsilon\}$$

دقت کنید که مرتبه زمانی محاسبه فرمول بالا نمایی است و نمیتواند به طور بهینه تعداد ایتریشن ها را تشخیص دهد چرا که حالت های پایه و نقطه شروع محاسبات دو عبارت $G^{(0:k-1)}$ و $G^{(0:k)}$ با یکدیگر تفاوت داشته و در نتیجه وضعیت های میانی آنها هم یکسان نبوده پس قابلیت استفاده مجدد ندارند.

در نتیجه برای محاسبه تک تک عبارات $G^{(0:k+i)}$ به ازای i های مختلف نیاز به محاسبه مجدد از پایه داشته که باعث رشد نمایی حجم محاسبات میشود.

پس لازم است که از رویکرد دیگری برای تخمین بهینه تعداد ایتریشن ها استفاده کنیم، مشخص کردن ایتریشنی که تفاوت دریافت اطلاعات ناشی از گسترش و بررسی بیشتر گراف نسبت به ایتریشن بعدی آن قابل چشم پوشی باشد میتواند سیگنال مناسبی باشد. به عبارت دیگر میتوان اینگونه تفسیر کرد که اطلاعات هر گره با استفاده از مدل انتقال پیام ما تا چه عمقی توانایی پخش شدن دارند. برای این مهم ما از رابطه تلپورت چشم پوشی کرده و تنها از رابطه پخش و پردازش گراف استفاده میکنیم که مارا به فرمول زیر میرساند.

$$E^{(l+1)} = \Phi(\alpha_{l+1}\tilde{A}E^{(l)}W) \quad (10)$$

در این فرمول وضعیت ابتدایی $E^{(0)}$ را برابر با $f_\theta(U)$ قرار میدهیم. دقت کنید که ما از α_{l+1} به عنوان اولین ضریبی که در ماتریس تلپورت $f_\theta(U)$ ضرب میشود. همگرایی فرمول (۱۰) مشابه با فرمول (۸) ثابت میشود.

۱-۴- بهینه سازی موثر

استفاده از فرمول (۷) در بهینه سازی اتوگراد همچنان منجر به مصرف بیش از حد حافظه خواهد شد چرا که مرتبه فضایی این رابطه خطی بوده و وابسته به تعداد لایه ها است. همچنین در forward pass گرادیان برای گره های دورتر به صفر همگرا میشود. با محاسبات گرادیان را تا جایی ادامه میدهیم که شرط همگرایی مشابه با بخش قبل حاصل شود. به دلیل همگرایی سریعتر در backward pass ، مصرف حافظه مدل کمتر شده و بهینه سازی مناسبی تلقی میشود. ما همچنان تعداد ایتریشن ها را محدود کرده تا جایی که ضمانت کنیم پیچیدگی فضایی به صورت ضریبی ثابت حاصل شده و فارغ از تعداد لایه ها یا پارامتر های دیگر باشد.

برای محاسبات مشتق های جزئی از فرمول (۸) استفاده مکنیم. ما نماد های جدیدی نیز معرفی میکنیم که شامل $Y' = f_\theta(G^{(0)})$ به عنوان خروجی مدل، Y به عنوان خروجی موردنظر و هدف، $L = l(Y', Y)$ به عنوان تابع خطا که تابعی مشتق پذیر و پیوسته است. توجه ما نیز به مشتق های جزئي تابع خطای ما نسبت به پارامتر های W و وضعیت ورودی B معطوف است. ما اجازه میدهیم که اتوگراد آماده ی ما مشتق جزئي $\frac{\partial L}{\partial G^{(0)}}$ محاسبه کند و سپس قانون مشتق زنجیره ای را برای محاسبه ی دیگر مشتق های جزئي به کار میبریم. برای نگارش آسان تر در اجرای قانون زنجیره ای لازم است نماد های دیگری نیز معرفی کنیم، $Z^{(n)} = \Phi(\alpha_n \tilde{A} G^{(n+1)} W + B)$ و $G^{(n)} = \Phi(Z^{(n)})$. تمامی مشتق های زنجیره ای اکنون توسط ۴ فرمول زیر قابل محاسبه هستند:

$$\frac{\partial L}{\partial \mathbf{G}^{(n)}} = \alpha_n \tilde{\mathbf{A}}^T \frac{\partial L}{\partial \mathbf{Z}^{(n-1)}} \mathbf{W}^T \tag{12}$$

$$\frac{\partial L}{\partial \mathbf{Z}^{(n)}} = \phi' \left(\alpha_n \tilde{\mathbf{A}} \mathbf{G}^{(n+1)} \mathbf{W} + \mathbf{B} \right) \odot \frac{\partial L}{\partial \mathbf{G}^{(n)}} \tag{13}$$

$$\frac{\partial L}{\partial \mathbf{W}} = \sum_{n=0}^{\infty} \alpha_n \left(\tilde{\mathbf{A}} \mathbf{G}^{(n+1)} \right)^T \frac{\partial L}{\partial \mathbf{Z}^{(n)}} \tag{14}$$

$$\frac{\partial L}{\partial \mathbf{B}} = \sum_{n=0}^{\infty} \frac{\partial L}{\partial \mathbf{Z}^{(n)}} \tag{15}$$

مشتق های جزئی $\frac{\partial L}{\partial W}$ و $\frac{\partial L}{\partial B}$ به دلایل مشابه با همگرایی فرمول (۸) و (۱۱)، نیز همگرا خواهند شد و آنقدر ایتریشن ها را اجرا خواهیم کرد تا شرط همگرایی برقرار شود. نرخ همگرایی برای فرمولی که از backward حاصل شده نیز در عمل از فرمول حاصل از forward pass سریعتر همگرا میشود که منجر به مصرف کمتر حافظه و بهینگی میشود. از نظر تئوری برای ضمانت کردن مصرف حافظه به صورت ثابت و مستقل از پارامتر، در تمامی جمع ها و سری های مشتق های جزئی تنها n عضو درنظر گرفته میشود، بهینگی این موضوع در تحقیق the effectiveness of Truncated Backpropagation Through Time یا به اختصار TBPTT بری داده های دنباله ای نشان داده شده است. همچنین این محدودیت منجر میشود پیچیدگی زمانی backward پس نیز ثابت باشد. بر اساس میزان حافظه موجود و در دسترس، ما وضعیت های

محاســـبات میانی مشـــتق های جزئي را به عنوان Gradient Checkpoint درنظر میگیریم تا مصرف حافظه را نیز بیش از بیش کاهش دهیم. دقت کنید که استفاده از وضعیت های ذخیره شده میانی، ما قادر هستیم تمامی اجزای مورد نیاز برای محاسبات بیشتر را بازسازی کنیم. در اصل میتوان بار حافظه ای نگهداری وضـــعیت های مشـــتق های جزئی را با تعداد کمی forward pass جبران کرده و تعداد بســـیار کمتری از وضـــعیت ها را ذخیره کنیم. دقت کنید که در مرحله backward پاسخ نهایی تمامی $G^{(0)}, \dots, G^{(n)}$ ببه طور صـریح نیاز اسـت. همچنین ما همگرایی تمامی $G^{(i)}$ را با اســـتفاده از این واقعیت که $G^{(0;k)} < G^{(n;k)}$ و در نتیجه با افزایش کمی تعداد ایتریشن $G^{(0;k+n)}$ در forward pass اولیه ضمانت میکنیم.

۵. فصل پنجم

آزمایشات

در این بخش مدل PPRGNN را روی دیتاســت های مختلفی ارزیابی کرده اند و نتایج را با ســایر مدل ها و شـبکه هایی که به صـورت عمق بی نهایت هسـتند مقایسـه کرده اند. برای مثال در جدول زیر ویژگی های دیتاست های استفاده شده، آورده شده است:

Dataset	# of Graphs	Avg. # of nodes	# of classes
Amazon	1	334 863	58
PPI	22	2373	121
MUTAG	188	17.9	2
PTC	344	25.5	2
COX2	467	41.2	2
PROTEINS	1113	39.1	2
NCI1	4110	29.8	2

در واقع این مدل را روی تسک های مختلف در حوزه گراف بررسی کرده اند:

:inductive node classification

در این نوع تسک ها، مدل برچسب گره هایی را پیش بینی می کند که در طی فرایند یادگیری آنها را ندیده است.

:Transductive node classification
در این نوع تسک ها، مدل برچسب گره هایی را پیش بینی می کند که قسمتی از گرافی بوده اند که مدل روی آن آموزش دیده است. به عبارتی مدل feature های گره هایی که قرار است برچسب آنها را پیش بینی کند در فرایند آموزش می بیند یا بهتر است بگوییم ساختار کل گراف را یاد می گیرد و تنها برچسب برخی از آنها را نمی داند.

:Graph classification
در این نوع تسک ها، مدل کل گراف ها را به دسته بندی های مختلف نظیر می کند.

به طور مثال، در دیتاست PPI(Protein-Protein interactions)، تسک inductive node classification مورد نظر است. این دیتاست، مجموعه داده ای از پروتئین ها به عنوان گره های گراف و ارتباطات بین آنها است. تسک، تعیین نقش(کلاس) پروتئین ها در کل گراف است. در این ارزیابی، ۱۸ گراف از دیتاست برای آموزش، ۲ گراف برای validation و ۲ گراف هم برای آزمایش درنظر گرفته شده اند. نتایج مربوط به مدل های مختلف روی این مجموعه داده در جدول زیر آمده است:

Method	Micro-F_1-Score
MLP	46.2
GCN	59.2
SSE	83.6
GAT	97.3
IGNN	97.6
APPNP	44.8
PPRGNN	**98.9**

در مقاله بیان شده که مدل PPRGNN بیش تر از ۵۰ درصد خطا را در مقایسه با مدل IGNN کاهش می دهد. همچنین PPRGNN آموزش داده شده روی این دیتاست، از ۸۲ تکرار message passing استفاده می کند.

مدل های دیگر مانند GAT و GCN پایه از حداکثر ۳ تکرار استفاده می کنند. در واقع این موضوع به این دلیل است که PPRGNN در عین اینکه دچار over-smoothing نمی شود، می تواند تا حد خوبی اطلاعات کل گراف را بدست بیاورد و اطلاعات گراف را تا عمق زیادی بیرون بکشد.

همین طور مقایسه دقیق تری میان PPRGNN و IGNN انجام شده است که نتایج به صورت زیر است:

Dataset	Method	Epochs	Avg. Time per Epoch	Total Time
Amazon (0.05)	IGNN	872	14s	3h 21m
	PPRGNN	**175**	**11s**	**32m**
PPI	IGNN	58	26s	25m
	PPRGNN	**47**	**18s**	**14m**

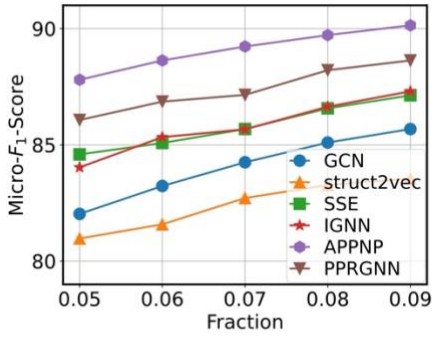
مدل PPRGNN معرفی شده در مقاله، به تکرار های کمتری نیاز دارد و همچنین زمان مورد نیاز برای هر epoch هم در آن کمتر است. در واقع سریعتر همگرا می شود. دلایل آن هم استفاده نکردن از Projected Gradient Descent و کنترل کردن سرعت همگرایی با پارامتر ε است. همان طور که در فصول قبلی دیدیم، Projected Gradient Descent می تواند بد تر از gradient descent معمولی عمل کند و کارآمدی کمتری داشته باشد. همچنین مدل APPNP به دلیل فیکس درنظر گرفتن شانس تلپورت، عملکرد خوبی روی دیتاست PPI نداشته است و underfit رخ داده است.

در بخش بعدی، دیتاست Amazon مورد بررسی قرار گرفته است. این دیتاست شامل گرافی از محصولات آمازون است بدین شکل که محصولات گره های گراف هستند و محصولاتی که که با هم خریداری شده اند، با یک یال به یکدیگر متصل شده اند. این گراف، گراف بسیار بزرگی درنظر گرفته می شود که شامل ۳۳۴۸۶۳ گره و ۹۲۵۸۷۲ یال است. تسک، پیش بینی کردن نوع گره ها(محصولات) است.

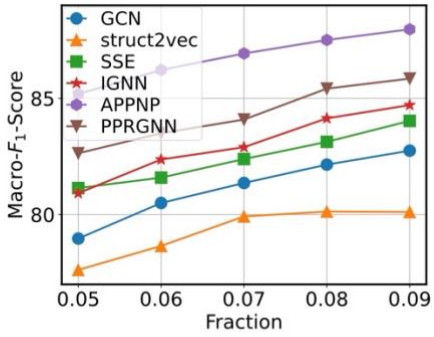
برای آموزش مدل، ۵ تا ۹ درصد گره ها برای آموزش درنظر گرفته می شوند.حال ۱۰ درصد از گره های آموزشی را به صورت فیکس برای آموزش درنظر می گیرند و سایر را برای Validation درنظر می گیرند.

در این ارزیابی، چالش اصلی پیچیدگی الگوی موجود در گراف نیست بلکه نامتوازن بودن بسیار زیاد دیتاست آمازون است.

مدل های مختلف را روی این دیتاست ارزیابی کرده اند که نتایج آنها را می توانیم در شکل زیر مشاهده کنیم:



(a) Micro-F_1-Scores.



(b) Macro-F_1-Scores.

همان طور که مشـــاهده می کنیم، تنها مدل APPNP بهتر از PPRGNN عمل می کند. دقت کنیم مدل APPNP پیچیدگی کمتری از مدل PPRGNN دارد و الگوهایی که کشف می کند هم پیچیدگی کمتری دارند. در این تسک نیز چون الگوی پیچیده ای در گراف وجود ندارد(طبق ادعای مقاله) و مسئله اصلی غیرمتوازن بودن کلاس های گره های است، نویسندگان توجیه کرده اند که استفاده از APPNP می تواند بهتر باشد و احتمال overfitting را بدلیل پیچیدگی کمتر، کاهش دهد.

همین طور روی این دیتاست نیز طبق شکل قبلی، مدل PPRGNN نسبت به مدل IGNN سریع تر همگرا می شود.

2-1- پیاده سازی

در این بخش ما سعی کردیم دو مدل PPRGNN و APPNP را روی دیتاست آمازون پیاده سازی کنیم و نتایج انها را با هم مقایسه کنیم.

کد در لینک زیر پیاده سازی شده است:

https://colab.research.google.com/drive/19ZpJVOgqyA74SkZH49a7uhSXQ2k0Jx5M

در این پیاده سازی، از متد های موجود در مخزن گیت هاب زیر که مربوط به مقاله مربوطه است استفاده شده است:

https://github.com/roth-andreas/pprgnn

همچنین دیتاست Amazon-all در درایو زیر قرار گرفته است:

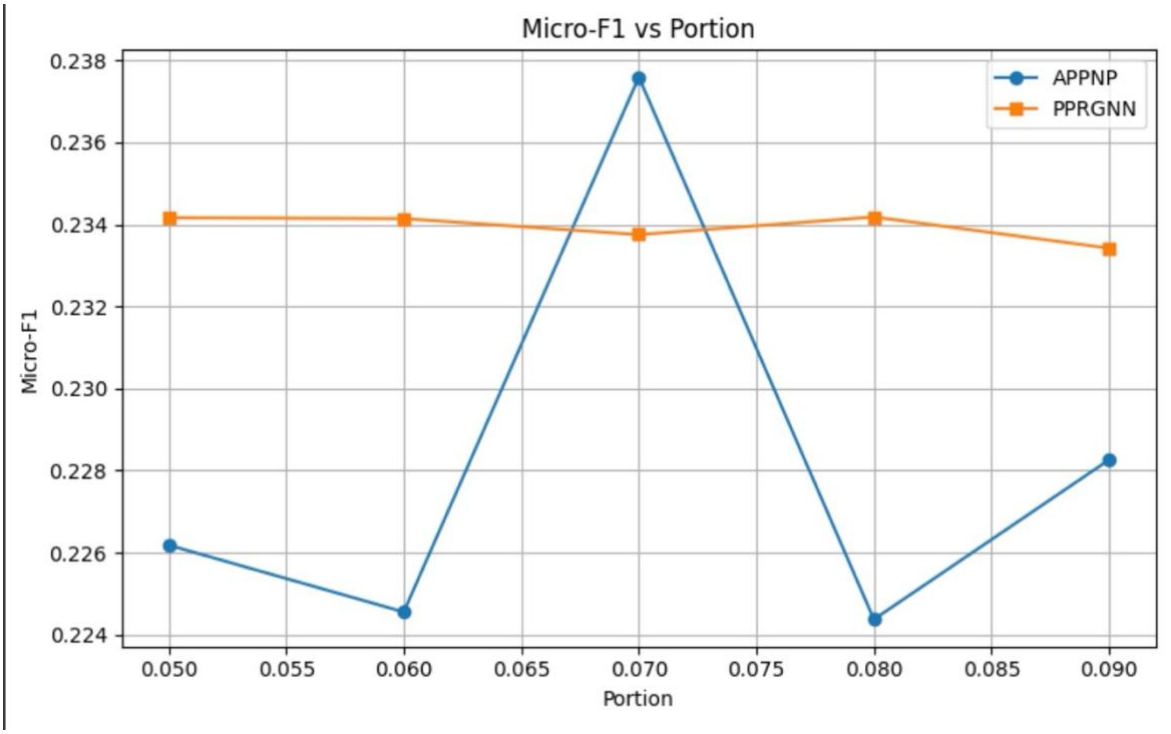
https://drive.google.com/drive/folders/1uS0mnNInHkmR3TL3JsD4gx5hKcNAIj6z

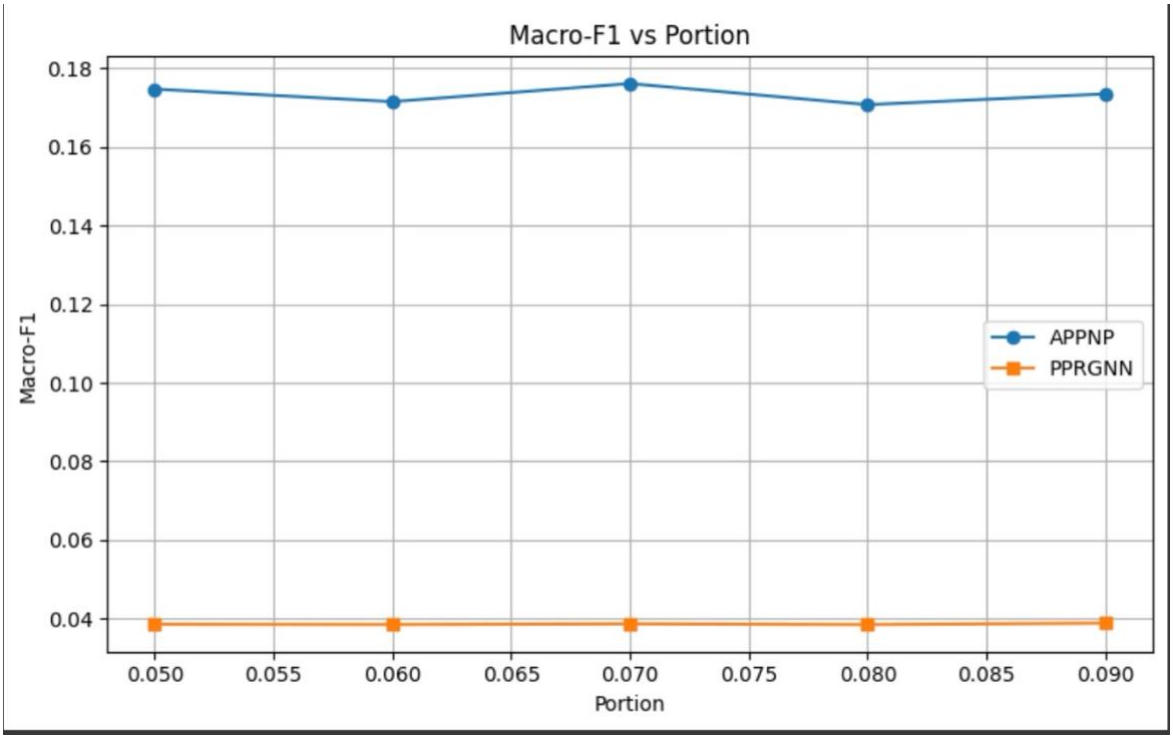
و قابل دانلود از لینک زیر است:

https://www.dropbox.com/sh/rjf9nlr94c7zmuk/AAAcCRLmgBNyp4aAyKcKHYi2a?dl=0

دیتای مربوط به نسبت های مختلف، به صورت فیکس در فایل ها قرار گرفته اند. در واقع دیتا از قبل نمونه گیری شده، در فایل ها قرار داده شده و در هنگام اجرای مدل ها صرفا از این فایل ها خوانده می شود.

در زیر نتایج پیاده سازی متد های موجود در مخزن بیان شده را به صورت نمودار می توانیم مشاهده کنیم:





نکته عجیب در اینجا متفاوت شدن نتیجه با آنچه در مقاله بیان شده است. تمامی کد ها کاملا منطبق با کد های موجود در مخزن است و پارامتر های مدل ها نیز به طور مشابه مقدار دهی شده اند.

اما نکته قابل توجه در این پیاده سازی تابع Evaluation آن است. در تابع Evaluation ابتدا برای هر سمپل، کلاس های مربوطه به آن استخراج می شود(دقت کنیم لیبل این گونه در نظر گرفته شده که هر سـپمل برداری از ۰ ها و ۱ ها اسـت یعنی هر کالا در این دیتاسـت می تواند چند تگ(برچسب) خورده باشد). بعد از بدست آوردن تعداد تگ های هر نود، سپس به تعداد این تگ ها بیش ترین امتیاز هایی که مدل پیش بینی کرده است بدست آورده می شود و در ایندکس آنها در بردار پیش بینی ۱ قرار داده می شود. سپس مکان ۱ های پیش بینی با ۱ های اصلی در دیتاست مقایسه می شود.

همان طور که مشخص است این یک نوع تقلب(cheating) محسوب می شود. زیرا فرض بر این است که ما هیچ گونه اطلاعاتی از داده های تست خود نداریم در حالی که در این تابع تعداد تگ های هر سمپل را می دانیم. حتی می توان به عنوان leakage هم این موضوع را درنظر گرفت.

نکته دیگر این است که اصلا در دیتاست آمازون همانطور که در پیپر بیان شده، تسک پیش بینی نوع هر کالا(کلاس هر نود) است و تسک mult-label classification نیست. با این حال با فرض

جمع آوری دیتا به شکلی دیگر و چند لیبل بودن نود ها، منطق برداشتن k امیتاز بیش تر از بردار پیش بینی هم نشـان می دهد ممکن اسـت مدل برای کلاس های دیگر احتمال بالایی در نظر بگیرد که نشـان دهنده عملکرد نه چندان خوب مدل اسـت. در کل پیاده سـازی تابع Evaluation با چنین منطق نادرسـت و زیرکانه ای، منجر به گرفتن نتایج کاملا متفاوت با آنچه در مقاله ذکر شـده بود، شده است.(البته درصورت صحیح بودن نتایج ذکر شده در مقاله)

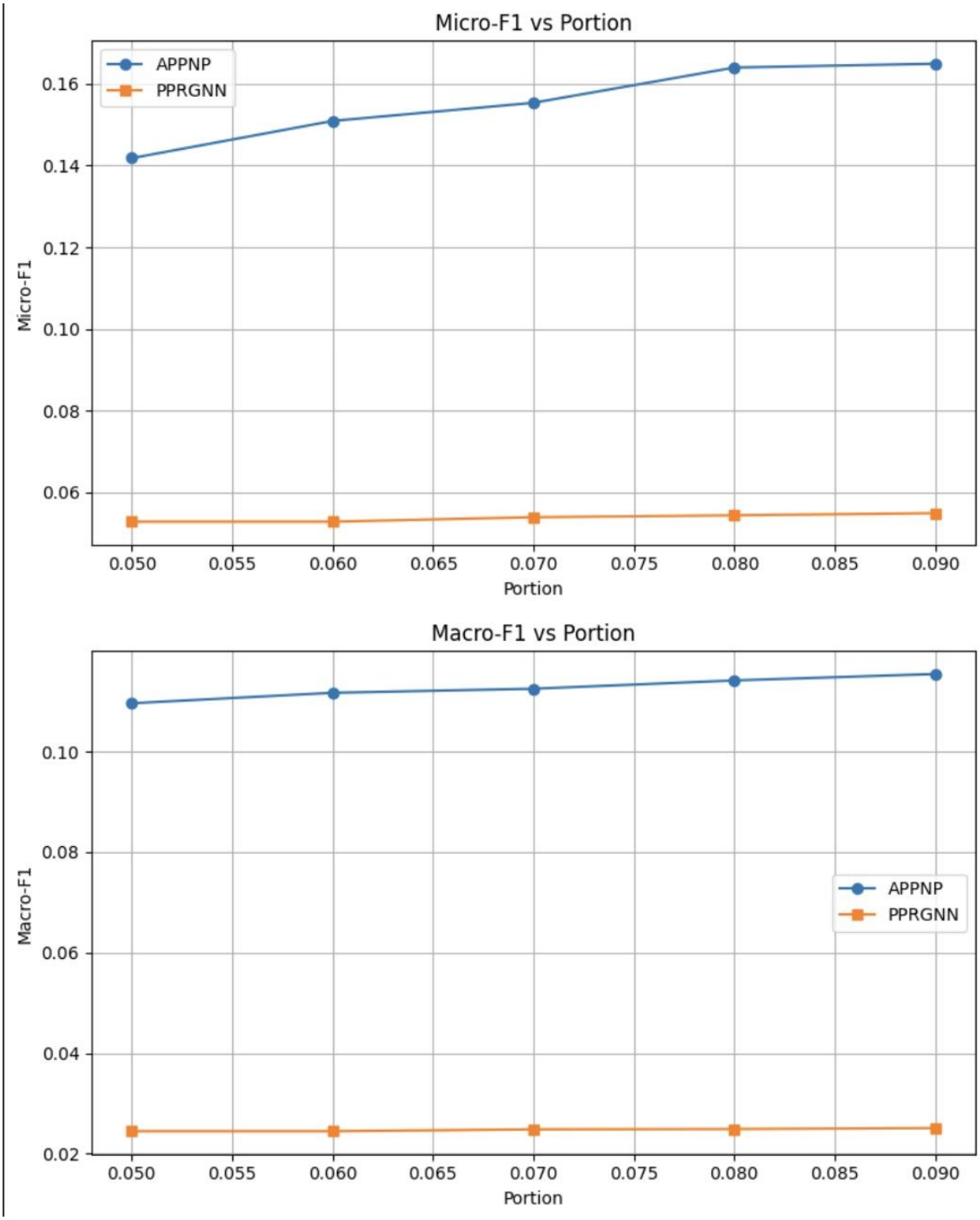
در ادامه تابع Evaluation جدیدی تعریف کردیم که منطق ساده ای نیز دارد. با درنظر گرفتن آستانه ای، در صورتی که احتمال بدست آمده برای کلاسی در بردار پیش بینی از مقدار آستانه بیش تر بود، مقدار ۱ و در غیر این صورت مقدار ۰ در بردار ست می شود. در نهایت با مقایسه ۱ ها در بردار پیش بینی و بردار اصلی نمونه، معیار های micro f1 و macro f1 محاسبه می شوند. بدون دانستن اطلاعاتی از نمونه های تست.

نحوه محاسبه آنها نیز به شکل زیر است:

$$F1_{micro} = \frac{2 \times \text{Precision}_{micro} \times \text{Recall}_{micro}}{\text{Precision}_{micro} + \text{Recall}_{micro}}$$

$$F1_{macro} = \frac{1}{C} \sum_{c=1}^{C} F1_{class_c}$$

در ادامه هم نتایج بر اساس تابع Evaluation جدید به شکل زیر درمی آید:



که در صورت صحیح بودن نتایج ذکر شده در مقاله، در پیاده سازی جدید، نتایج تطابق بیش تری دارند.

نتیجه گیری

در این مقاله با شبکه عصبی گرافی به نام PPRGNN آشنا شدیم که محدودیتی روی عمق ندارد و دچار بیش هموارسازی نمی شود. پایه ایده آن هم، ایده الگوریتم personalized pagerank است. طبق ادعای مقاله، این مدل تقریبا در تمامی دیتاست ها، از مدل های قابل مقایسه دیگر بهتر عمل می کند. در پیاده سازی روی دیتاست آمازون هم عملکرد دو مدل PPRGNN و APPNP را مشاهده کردیم و نتایج را با آنچه در مقاله ذکر شده بود مقایسه کردیم که در ابتدا نتایج کاملا متفاوت بودند اما با اصلاح پیاده سازی، نتایج تا حد خوبی به نتایج موجود در مقاله نزدیک شد. همچنین دریافتیم می توان با بهینه سازی روی تعداد لایه، پیچیدگی زمانی و حافظه مدل را بهینه کرد. در کل در هرتسکی که به یک مدل گرافی با لایه های message passing نیاز داریم، PPRGNN یک مدل کاندید برای انتخاب خواهد بود.