به نام خدا

دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

درس تحلیل شبکه‌های پیچیده

استاد حقیرچهرقانی

پروژه پایانی

علیرضا مازوچی

۴۰۰۱۳۱۰۷۵

سوال ۱

الف) متناسب با اصلاحیه مجموعه‌داده PubMed مورد استفاده من برای سوال ۱ بوده است. این مجموعه‌داده دارای ویژگی‌های زیر است:

1. تنها شامل یک گراف است.
2. دارای ۱۹۷۱۷ گره است.
3. هر گره ۵۰۰ ویژگی دارد.
4. دارای ۸۸۶۴۸ یال است.
5. یال‌ها بدون ویژگی هستند.
6. هر گره متعلق به یکی از سه کلاس ۰، ۱ و یا ۲ است.
7. از بین گره‌های موجود ۶۰ تا برای بخش آموزش، ۵۰۰ تا برای بخش اعتبارسنجی و نهایتا ۱۰۰۰ گره برای بخش آزمون در نظر گرفته شده است.

ب) شبکه GCN

به نظر می‌رسد این قسمت دو بخش اصلی داشته باشد؛ ابتدا یک شبکه ثابت با دو لایه GCN و یک لایه تماما متصل ایجاد کرده‌ام و سپس صحت چندین تنظیم مختلف از تعداد ویژگی‌های دو لایه مخفی را محاسبه کرده‌ام. برای آنکه نتایج قابل اعتمادتر باشد برای هر تنظیم چهار مرتبه اجرا گرفتم و میانگین صحت‌ها را به عنوان صحت نهایی اعلام کردم. همچنین از مکانیسم Early Stopping برای جلوگیری از بیش‌برازش‌شدن مدل بهره گرفتم. در جدول ۱ نتایج این قسمت آورده شده است. مطابق این جدول ساده‌ترین تنظیم من با دو لایه GCN مخفی با تعداد ویژگی خروجی ۱۶ و ۸ بهترین نتایج را داشته است؛ اگر چه سایر تنظیم‌ها هم فاصله چندانی با آن نداشته‌اند.

جدول ۱ – نتایج قسمت اول شبکه GCN (بررسی تعداد ویژگی مناسب GCN)

اندازه لایه‌های مخفی	صحت آزمون
۱۶ و ۸	۷۵.۸۰٪
۳۲ و ۱۶	۷۴.۷۵٪
۶۴ و ۳۲	۷۵.۰۵٪
۲۵۶ و ۶۴	۷۵.۰۲٪
۱۰۲۴ و ۱۰۲۴	۷۵.۲۰٪

برای قسمت دوم این بخش مطابق درخواست سوال نه حالت (سه حالت برای تعداد لایه GVN و سه حالت برای تعداد لایه تماما متصل) را در نظر گرفتم. شرایط آموزش مشابه قسمت قبل است. از آنجایی که ترکیبات ساده صحت مناسبی داشتند، برای این قسمت هم شبکه‌ها را تعداد ویژگی کم آموزش دادم. نتایج این قسمت در جدول ۲ ارائه شده است.

جدول ۲ – نتایج قسمت دوم شبکه GCN (بررسی تعداد لایه مناسب GCN و تماما متصل)

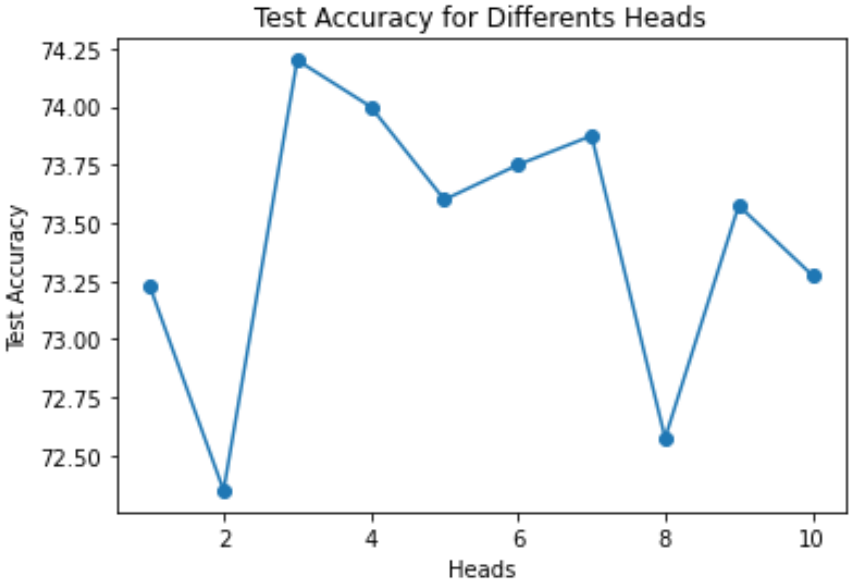
تعداد لایه‌های GCN	تعداد لایه‌های تماما متصل	صحت آزمون
۱	۱	۷۰.۰۰٪
۲	۱	۷۲.۶۲٪
۳	۱	۷۵.۹۵٪
۱	۲	۴۹.۵۰٪
۲	۲	۵۳.۵۰٪
۳	۲	۷۱.۳۵٪
۱	۳	۴۱.۴۲٪
۲	۳	۵۷.۳۰٪
۳	۳	۶۵.۷۲٪

برخلاف آزمایش پیشین صحت‌های بدست آمده بسیار متفاوت و قابل تامل است. می‌توان دید که افزایش لایه‌های GCN همواره باعث بهبود دقت شده است. البته این مورد تنها تا سه لایه برقرار است و معلوم نیست افزایش بیشتر تا کجا ارزشمند

باشد. مورد دیگری که می‌توان دید آن است که افزایش تعداد لایه‌های تماما متصل باعث کاهش صحت شده است. احتمالا لایه‌های GCN می‌توانند نمایش خیلی مناسبی را ارائه دهند که با الگوهای خطی قابل جداسازی است. در این شرایط افزایش لایه‌های تماما متصل به دلیل افزایش پیچیدگی جز با جایگزینی لایه‌های GCN امکان‌پذیر نیست. در اینجا شایان ذکر است که لایه‌های GCN قادرند تا نمایشی مبتنی بر ساختار گراف ارائه دهند در حالی که لایه‌های تماما متصل از چنین قابلیتی برخوردار نیستند. به علاوه آنکه لایه‌های تماما متصل تعداد پارامتر زیادی را می‌طلبد. پس در مجموع و در این شرایط بهتر است که پیچیدگی مدل به واسطه افزودن لایه‌های GCN و نه لایه‌های تماما متصل انجام گیرد. در مورد تعداد مطلوب ویژگی‌های لایه‌های مخفی هم در قسمت قبل توضیح دادیم که به نظر می‌آید تعداد ویژگی کم به اندازه کافی مناسب باشد.

شبکه GAT

برای بخش اول این بخش، شبکه‌ای شامل سه لایه GAT با تنظیم یکسان در نظر گرفتم و تعداد سرهای توجه را تغییر دادم که نتایج مطابق نمودار ۱ بدست آمد. بر اساس این نمودار به نظر می‌رسد که ۳ سر بهترین نتیجه را داشته است.



تصویر ۱ – تاثیر تعداد سر بر روی صحت آزمون

برای بخش دوم نیاز به تغییر پایه‌ای لایه GATConv موجود بود. فایل کلاس موجود در کتابخانه[1]را عینا کپی کردم و صرفا تغییرات جزئی در آن اعمال کردم. در لایه پیاده‌سازی‌شده آرگومانی تحت عنوان concat وجود داشت؛ اگر مقدار آن True بود، خروجی هر سر با سایر سر ها ادغام می‌شود و در غیر این صورت میانگین‌گیری انجام می‌شود. ما باید لایه میانگین‌گیری را به گونه‌ای تغییر می‌دادیم که هر سر وزن قابل آموزش مجزایی از سایر سرها داشته باشد. با کمک یک CNN یک بعدی و با اندازه هسته ۱ خیلی ساده می‌توان یک لایه میانگین‌گیر با وزن‌های قابل آموزش را پیاده‌سازی کرد[2]. این تغییر در بخش forward کلاس موجود اعمال شده است.

صحت برای شبکه جدید مبتنی بر لایه سفارشی‌شده به ۷۴.۵۰٪ می‌رسد. در اینجا بد نیست که اشاره کنم برای بدست آوردن این دقت تعداد لایه‌های GAT را کاهش دادم؛ چراکه با شبکه‌ای مشابه با GAT استاندارد صحت اصلا خوب نبود.

شبکه GAT V2

برای این قسمت هم مشابه با بخش دوم GAT کد کلاس موجود در کتابخانه را کپی کردم[3] و تغییراتی بر آن اعمال. در تابع کد موجود عمل جمع h_i و h_j در حال انجام است. به نظر می‌رسد این عمل در پیاده‌سازی به جای ادغام مورد استفاده است. چراکه سایر محاسبات کاملا منطبق است. لذا یک تابع به نام message_operation به پیاده‌سازی موجود افزودم که کار آن اعمال هر نوع تابعی بر h_i و h_j است.

در جدول ۳ نتایج حالت استاندارد و سه حالت افزونه آورده شده است. اگرچه تفاوت فاحش نیست ولی به نظر می‌رسد همان حالت استاندارد حداقل برای شبکه و مجموعه‌داده من بهترین است.

---

[1] https://github.com/pyg-team/pytorch_geometric/blob/master/torch_geometric/nn/conv/gat_conv.py

[2] https://stackoverflow.com/a/58574603/8961642

[3] https://github.com/pyg-team/pytorch_geometric/blob/master/torch_geometric/nn/conv/gatv2_conv.py

جدول ۳ – نتایج GAT V2 و افزونه‌هایش

مدل	صحت آزمون
استاندارد	۷۵.۲۲٪
مینیمم	۷۴.۵۵٪
ماکسیمم	۷۴.۷۰٪
آدامار	۷۴.۳۵٪

ج) در جدول ۴ خلاصه‌ای از بهترین مدل‌ها آورده شده است. همانطور که قابل مشاهده است با انواع مدل‌ها به دقت‌های تقریبا یکسانی می‌توان رسید. در این شرایط بیان برتری یک مدل چندان صحیح به نظر نمی‌رسد و برتری GCN احتمالا به دلیل آزمایشات متعدد درخواست‌شده در صورت سوال بوده است.

جدول ۴ – خلاصه نتایج بهترین مدل‌ها

مدل	صحت آزمون
GCN	۷۵.۹۵٪
GAT	۷۴.۲۵٪
GAT + Learnable Weight	۷۴.۵۰٪
GAT V2	۷۵.۲۲٪

سوال ۲

الف) برای این سوال من مقاله Simplifying Graph Convolutional Networks را انتخاب کردم. ابتدا ایده اصلی مقاله را تشریح می‌کنم.

در مقاله ادعا شده است که مدل‌های GCN و افزونه‌های آن برای نمایش گراف شدیدا مورد توجه هست. از آنجایی که ایده اصلی آن از مدل‌های مبتنی بر یادگیری عمیق بدست آمده است، طبیعتا پیچیدگی و محاسبات بالایی دارد. آن‌ها تاکید دارند که در اغلب مسائل دسته‌بندی دنیای واقعی با یک مسئله خطی ساده مواجه هستیم و شاید پیچیدگی زیاد که در مدل‌های عمیق استفاده شده است لازم نباشد؛ به بیان دیگر همواره باید از مدل ساده استفاده کرد مگر آنکه برای مسئله نیاز به پیچیدگی بیشتری وجود داشته باشد.

ایده کلیدی مقاله از اینجا شروع می‌شود که GCN نسخه پیچیده‌تر از یک مدل ساده موجود نبوده است! بلکه مستقیما از سایر حوزه‌ها وارد حوزه گراف شده است. در این مقاله در جهت عکس تلاش برای ارائه یک نسخه ساده از GCN شده است. برای ساده‌تر کردن مدل لایه‌های غیرخطی میان لایه‌های کانوولوشنی میان GCN را حذف و به تبع لایه‌های باقی مانده را ترکیب کرده‌اند تا نهایتا یک لایه خطی ساده باقی بماند. آن‌ها تنها به یک لایه غیرخطی به عنوان لایه دسته‌بندی نیاز دارند. مدل ساده باقی مانده را Simple Graph Convolutional Network یا به اختصار SGC می‌نامند.

نویسندگان مقاله مذکور مدعی هستند که چنین ساده کردنی دقت مدل را کاهش قابل توجه نمی‌دهد و حتی در مواردی باعث بهبود دقت می‌شود. اما در مقابل کاهش چشمگیر زمان اجرا منجر می‌شود. کاهش زمان تسهیل‌گر آموزش مدل بر روی مجموعه‌داده‌های بزرگ خواهد بود.

حال بیاییم و مدل SGC را دقیق‌تر بررسی کنیم؛ فرض کنید X ماتریس ویژگی‌های اولیه گره‌ها، $H^{(k)}$ ماتریس نمایش گره‌ها بعد از لایه‌ی k ام، A ماتریس همسایگی باشد. یک نسخه از نرمال‌شده از ماتریس همسایگی که دارای خود حلقه (self-loop) است معمولا در شبکه‌های GCN با نماد S مورد استفاده است که فرمول آن عبارت است از:

$$S = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$$

در این رابطه داریم: $\tilde{A} = A + I$ و $\widetilde{D}$ ماتریس درجه $\tilde{A}$ خواهد بود. بر اســـاس ماتریس Sو نمایش گره‌های همسایه مطابق رابطه زیر نمایش جدید یک گره را می‌سازند:

$$\overline{H}^{(k)} \leftarrow SH^{(k-1)}$$

در مدل‌های GCN که لایه غیرخطی وجود دارد، نهایتا مطابق با رابطه زیر نمایش یک گره برای لایه k بدست می‌آید:

$$H^{(k)} = ReLU\left(\overline{H}^{(k)} * \Theta^{(k)}\right)$$

اما همانطور که پیش‌تر توضیح داده شد، برای SGC خبری از لایه‌های غیرخطی نیست و عملا می‌توان تمام ماتریس‌های دیگر یعنی S و $\Theta^{(k)}$ را در هم ضرب و ترکیب کرد. در اینجا شایان ذکر است که $H^{(0)} = X$. پس نهایتا رابطه SGC با احتساب لایه دسته‌بندی نهایی به نحو زیر بدست می‌آید:

$$\hat{Y} = softmax\left(S \dots SSX\Theta^{(0)}\Theta^{(1)} \dots \Theta^{(K)}\right) = softmax(S^K X\Theta)$$

پر واضح است که می‌توان $S^K X$ را یک بار محاسبه کرد و Θ را به عنوان پارامترهای مسئله یاد گرفت. با این توضیحات جای تعجب ندارد که سرعت مدل بسیار بهبود پیدا کند.