دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر

پروژه کارشناسی

بررسی عملکرد شبکه‌های عصبی گرافی در رگرسیون گره‌ها

نگارش

امیرمهدی زرین‌نژاد

استاد راهنما

جناب آقای دکتر مصطفی حقیرچهرقانی

مهر ۱۴۰۲


بسم الله الرحمن الرحيم

تقدیم به پدر، مادر و برادر دلسوز و مهربانم به سبب اینکه، همیشه در مراحل زندگی من را یاری نموده اند و به حمایت از من پرداخته اند.

سپاس‌گزاری

از استاد دلسوز و محترم؛ جناب آقای دکتر مصطفی حقیر چهرقانی که با صبر و حوصله، از هیچ کمکی در مسیر انجام این پروژه و نوشتن این پایان‌نامه از من دریغ ننمودند؛ کمال تشکر و قدردانی را دارم.

امیرمهدی زرین‌نژاد

مهر ۱۴۰۲

چکیده

شبکه‌های عصبی گرافی، یکی از حوزه‌های پر رونق در علم داده‌ها و هوش مصنوعی هستند. این شبکه‌ها طراحی شده‌اند تا بتوانند داده‌هایی با ساختار گرافی مانند شبکه‌های اجتماعی، شبکه‌های مخابراتی، یا ساختارهای مولکولی را تحلیل و پردازش کنند. رگرسیون یکی از مسائل پرکاربرد و مهم در زمینه هوش مصنوعی با شبکه عصبی گرافی است که امکان پیش‌بینی و تخمین مقادیر پیوسته برای گره‌ها در یک گراف را ممکن می‌سازد. با استفاده از شبکه‌های عصبی گرافی، می‌توانیم بهبود قابل توجهی در دقت و کارایی مدل‌ها در پیش‌بینی مقادیر پیوسته داشته باشیم. در این پروژه، مقایسه و بررسی عملکرد مدل‌های مختلف شبکه عصبی گرافی در مسئله رگرسیون گره‌ها مورد بررسی قرار گرفت. در ابتدا، یک مجموعه دادگان ساختاریافته گرافی در حوزه شبکه ارجاعات تهیه و آماده‌سازی شد. سپس چندین مدل شبکه عصبی گرافی با تنظیمات مختلف ایجاد و بر روی مجموعه دادگان اجرا شد. این مدل‌ها برای پیش‌بینی مقادیر گره‌ها به کار گرفته شدند. معیارهای متعددی نظیر خطای میانگین مربعات، خطای میانگین مطلق و خطای جذر میانگین مربعات برای محاسبه خطای بین مقادیر پیش‌بینی شده با مقادیر واقعی گره‌ها و نهایتا برای ارزیابی مدل‌ها مورد استفاده قرار گرفت. تاثیر مقادیر مختلف متغیرهای مدل‌ها بر روی عملکردش نیز مورد بررسی قرار گرفت. نتایج نشان می‌دهد که شبکه‌های عصبی گرافی برای مسائل رگرسیون گره‌ها توانایی بسیار خوبی دارند. این پروژه به پیشنهاد مدل‌ها و پارامترهای مناسب برای مجموعه دادگان خاص اشاره می‌کند و عملکرد هرکدام را بیان می‌کند. این پروژه می‌تواند به توسعه مدل‌های پیش‌بینی در حوزه‌هایی نظیر شبکه‌های اجتماعی، شبکه ارجاعات، علوم زیستی، مهندسی و تجارت کمک کند.

واژه‌های کلیدی:

شبکه عصبی گرافی، رگرسیون گره‌ها، شبکه ارجاعات

فهرست مطالب

فهرست تصاویر

شکل صفحه

فهرست نمادها

نماد	مفهوم
G	گراف
V	مجموعه گره‌ها
n	تعداد گره‌ها
E	مجموعه یال‌ها
ε	خطای معدله
f	تابع
w_i	وزن مرتبط با ورودی i–ام
b	سوگیری(بایاس)
α	نرخ یادگیری
∇	گرادیان
σ	تابع فعال‌ساز
N	مجموعه گره‌های همسایه
$h_v^{(l)}$	بردار ویژگی گره v در لایه l
c_v	ثابت نرمال‌سازی

فصل اول

مقدمه

۱-۱ مقدمه

شبکه‌های عصبی گرافی(GNN)[۱] نوعی شبکه عصبی هستند که داده‌های با ساختار گراف[۲] را پردازش می‌کنند و در سال‌های اخیر به عنوان یکی از ابزارهای قدرتمند در حوزه هوش مصنوعی[۳]، یادگیری‌ماشین[۴] و داده‌کاوی[۵] مورد توجه فراوانی قرار گرفته‌اند. این شبکه‌ها را می‌توان برای کارهای پیش‌بینی در سطح گره، سطح ارتباطات و سطح کلی گراف استفاده کرد. به همین صورت، این شبکه‌ها با ترکیب قابلیت‌های شبکه‌های عصبی و قابلیت‌های مدل‌سازی گراف، توانسته‌اند به طور موفقیت‌آمیزی چالش‌های مختلفی را در حوزه‌های مختلفی از جمله تحلیل شبکه‌های اجتماعی، پیش‌بینی ترافیک و مسیریابی، پردازش زبان طبیعی، بیوانفورماتیک و بسیاری از مسائل دیگر به حل برسانند. رگرسیون[۶] یکی از مسائل پرکاربرد و مهم در حوزه هوش مصنوعی با شبکه عصبی گرافی است که امکان پیش‌بینی و تخمین مقادیر پیوسته[۷] برای گره‌ها(ویا دیگر بخش‌ها) در یک گراف را ممکن می‌سازد. با استفاده از شبکه‌های عصبی گرافی در رگرسیون، می‌توانیم از مزایای شبکه‌های عصبی معمولی و روش‌های سنتی بهره ببریم و نتایج قابل توجهی ارائه کنیم.

۱-۲ اهداف پروژه

در این پایان‌نامه، هدف ما مقایسه و بررسی عملکرد مدل‌های مختلف شبکه عصبی گرافی در مسائل کاربردی مختلف با محوریت رگرسیون است. قصد داریم مدل‌های مختلف شبکه عصبی گرافی را مقایسه و بررسی کرده و نقاط قوت و ضعف هریک از این مدل‌ها را شناسایی کنیم. با پیاده‌سازی و آموزش مدل‌ها بر روی مجموعه‌داده‌های عمومی، عملکرد آن‌ها ارزیابی کرده و با مقایسه نتایج بدست آمده (بین مدل‌های مورد بررسی و یا دربرابر روش‌های دیگر موجود در حوزه مورد) بهترین راهکار را برای مسئله مشخص کنیم. و نهایتا انگیزه ما بهبود روش‌های رگرسیون موجود و استفاده از توانایی‌های شبکه‌های عصبی گرافی در پیش‌بینی مقادیر پیوسته در گراف‌ها است. با استفاده از شبکه‌های عصبی گرافی، می‌توانیم بهبود قابل توجهی در دقت و کارایی رگرسیون بر روی گراف‌ها داشته باشیم و نتایج دقیق‌تر و قابل استنباط‌تری را ارائه دهیم. با مقایسه مدل‌های مختلف و پیاده‌سازی آن‌ها، می‌توان بهترین روش را برای هر مسئله مشخص کرده و در نتیجه دقت و کارایی مدل‌ها را با توجه به کاربرد بهبود بخشید. نهایتا با ترکیب ادبیات موجود و انجام ارزیابی‌های تجربی، امیدواریم بتوانیم به درک عمیق‌تر GNNها کمک کنیم و بینش‌هایی در مورد نقاط قوت و محدودیت‌های آن‌ها ارائه کنیم.

[1] Graph Neural Networks
[2] Graph
[3] Artificial Intelligence
[4] Machine Learning
[5] Data Mining
[6] Regression
[7] Continuous Values

۱-۳ ساختار پایان‌نامه

پایان‌نامه از شش فصل تشکیل شده‌است که ابتدا در فصل دو به توضیح ادبیات، اصطلاحات و مفاهیم مورد استفاده‌ی پروژه می‌پردازیم چراکه برای درک و تحلیل پروژه به این دانش نیاز داریم. در فصل سه فعالیت‌های پیشین و کارهای موجود در حوزه رگرسیون گره‌ها را معرفی و بررسی می‌کنیم. در فصل بعدی به معرفی مجموعه‌دادگان و اقدامات اولیه برای آماده‌سازی ساختار داده می‌پردازیم و اهمیت این امر را شرح می‌دهیم. در ادامه به شرح مدل‌های مورد استفاده و فعالیت‌ها علمی-عملی صورت گرفته می‌پردازیم و نهایتا ارزیابی و نتایج بدست‌آمده را توضیح می‌دهیم.

فصل دوم

ادبیات مسئله

۲-۱ مقدمه

توضیح و تشریح مفاهیم پایه یک بخش اساسی در توضیح پروژه است چراکه پایه و اساس دانش‌های بعدی را فراهم می‌کنند و درک مسئله و راه‌حل‌های ارائه شده را ممکن می‌سازند. در این فصل به توضیح مطالب مرتبط و پیش‌نیاز پروژه می‌پردازیم و درک مراحل بعدی را سهولت می‌بخشیم.

۲-۲ گراف

۲-۲-۱ تعریف

گراف [۱] ، یک مفهوم اساسی در ریاضیات و علوم کامپیوتر است که به مدل‌سازی ارتباطات بین داده‌ها و نقاط مختلف به کمک یال‌ها [۲] و گره‌ها [۳] می‌پردازد. گراف به عنوان یک داده‌ساختار اساسی در زمینه‌های مختلف از جمله علوم کامپیوتر، مهندسی شبکه، علوم اجتماعی، زبان‌شناسی، بیوانفورماتیک، و حتی مهندسی برق و مکانیک مورد استفاده قرار می‌گیرد. بنابراین گره‌ها موجودیت‌ها یا نقاط منفرد گراف و لبه‌ها [۴] (یال‌ها) اتصالات و روابط بین گره‌ها را بیان می‌کنند.

از نظر ریاضی، یک گراف G را می‌توان به صورت $G(V, E)$ نشان داد، که در آن V مجموعه‌ی گره‌ها و E مجموعه‌ی یال‌ها است. یال‌ها می‌توانند جهت‌دار (نشان‌دهنده رابطه یک‌طرفه) یا غیرجهت‌دار (نشان‌دهنده رابطه دوطرفه) باشند. علاوه بر این، لبه‌ها را می‌توان وزن داد، به این معنی که یک ارزش یا وزن به هرکدام نسبت داد به طوری که نشان‌دهنده یک ویژگی، فاصله یا هزینه خاص مرتبط با اتصال بین گره‌ها است.

در علوم کامپیوتر و ریاضیات، گراف‌ها را می‌توان براساس ویژگی‌ها و جزئیاتشان به انواع مختلفی دسته‌بندی کرد که داشتن درک درستی از آن‌ها برای انجام فعالیت‌های تحقیقاتی و اجرایی لازم است. در ادامه چند نوع رایج گراف‌ها آورده شده است:

- گراف‌های جهت‌دار ($Digraphs$): در این گراف‌ها یال‌ها جهت دارند. به عبارت دیگر، اگر یک یال از گره A به گره B وجود داشته‌باشد، این به معنای وجود یک یال از B به A نیست. این گراف‌ها اغلب به عنوان "گراف‌های جهت‌دار" یا "دیگراف‌ها" نیز شناخته می‌شوند.
- گراف‌های بدون جهت ($UndirectedGraphs$): در این گراف‌ها یال‌ها جهت ندارند. اگر یک یال بین گره A و گره B وجود داشته باشد، این به معنای وجود یک یال بین گره B و گره A نیز می‌باشد. اینها اغلب گراف‌های "بدون جهت" یا "ساده" نیز شناخته می‌شوند.

[1] Graph
[2] Link
[3] Node
[4] Edge

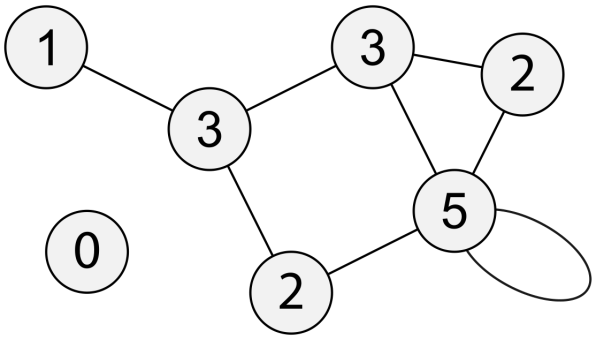
- گراف‌های وزن‌دار ($WeightedGraphs$): در این گراف‌ها، به هر یال یک وزن یا هزینه اختصاص داده می‌شود که معمولاً ویژگی‌های رابطه بین گره‌ها را نشان می‌دهد. از این وزن‌ها در مسائل بهینه‌سازی و تصمیم‌گیری استفاده می‌شود.

- گراف‌های چرخه‌ای و گراف‌های غیر چرخه‌ای: در گراف‌های چرخه‌ای، مسیرهایی وجود دارد که به گره شروع باز می‌گردند و چرخه ایجاد می‌کنند. در گراف‌های غیر چرخه‌ای، چنین چرخه‌ای وجود ندارد. گراف‌های غیر چرخه‌ای نیز به عنوان "درخت" شناخته می‌شوند و در ساختارهای سلسله‌مراتبی استفاده می‌شوند.

- گراف‌های منظم ($RegularGraphs$): در این گراف‌ها، همه گره‌ها دارای تعداد یال‌های یکسانی هستند. به عبارت دیگر، هر گره در یک گراف منظم دارای درجه یکسانی است. یک گراف کامل، که در آن هر جفت گره توسط یک یال به هم متصل می‌شوند، نمونه‌ای از یک گراف منظم است.

- گراف‌های کامل ($CompleteGraphs$): در گراف‌های کامل، هر جفت گره توسط یک یال به هم متصل می‌شوند و درواقع تمامی گره‌ها با یال‌ها به یکدیگر متصل هستند. تعداد کل یال‌ها در یک گراف کامل با n گره، برابر $n(n-1)/2$ است.

- گراف‌های دوبخشی ($BipartiteGraphs$): این گراف‌ها به دو مجموعه گره تقسیم می‌شوند و لبه‌ها فقط گره‌های مجموعه‌های مختلف را به هم متصل می‌کنند. گراف‌های دوبخشی در مسائلی مانند تخصیص منابع (به عنوان مثال، تخصیص وظایف به کارگران) استفاده می‌شود.



شکل ۲-۱: یک نمونه گراف بدون جهت دوبخشی با ۷ راس و ۸ یال

۲-۲-۲ تاریخچه

مفهوم گراف‌ها تاریخچه‌ای غنی دارد که آغاز به قرن هجدهم بازمی‌گردد، با مشارکت‌های قابل توجهی از ریاضیدانان و دانشمندان مختلف از جمله:

۱. لئونارد اویلر [۵] (۱۷۳۶) - تولد نظریه گراف: لئونارد اویلر به واسطه کارش بر روی مسئله هفت پل کونیگزبرگ، پایه‌گذاری نظریه گراف را انجام داد. او این مسئله را به شکلی که اکنون به عنوان یک گراف می‌شناسیم رسمیت داد و نشان داد که غیرقابل حل است. این مسئله و مقاله مشهور اویلر بیانگر تولد نظریه گراف بود.

۲. آرتور کیلی [۶] (۱۸۵۷) - نامگذاری گراف: آرتور کیلی با معرفی اصطلاح "گراف" و توسعه نماد گراف کمک مهمی به این حوزه کرد. اصطلاحات و نمادهایی که او معرفی کرد تا به امروز هم‌چنان به طور گسترده در نظریه گراف استفاده می‌شود.

۲-۲-۳ کاربرد

با توجه به ساختاری که گراف دارد، بسیاری از پدیده‌های جهان را می‌تواند مدل‌سازی کند و این موجب شده است که نظریه گراف در زمینه‌های مختلفی مانند علوم کامپیوتر(شبکه‌ها، الگوریتم‌های مسیریابی)، علوم اجتماعی(تحلیل شبکه‌های اجتماعی)، زبان شناسی(درخت نحو)، زیست شناسی(شبکه‌های تعامل پروتئین-پروتئین) و بسیاری موارد دیگر کاربرد داشته باشد.

۲-۳ رگرسیون

۲-۳-۱ تعریف

رگرسیون [۷] یک روش آماری است که برای قرن‌ها ابزاری بنیادی در درک و مدل‌سازی روابط بین متغیرها بوده است. رگرسیون به ما امکان می‌دهد تا تأثیر یک یا چند متغیر مستقل را بر یک متغیر وابسته تجزیه و تحلیل و کمی‌سازی کنیم. تحلیل رگرسیون چارچوبی را برای پیش‌بینی مقادیر، استنتاج علیت و درک الگوهای موجود در داده‌ها فراهم می‌کند. در این قسمت توضیح و تاریخچه‌ای از رگرسیون به همراه کاربردهای متنوع آن آورده شده است.

تجزیه و تحلیل رگرسیون در مرکزیت خود به دنبال ایجاد یک رابطه ریاضی بین یک متغیر وابسته ("Y") و یک یا چند متغیر مستقل ("X") است. ساده‌ترین شکل رگرسیون، رگرسیون خطی است که به صورت زیر بیان می شود:

$$Y = \beta_0 + \beta_1 X + \varepsilon \qquad (۱-۲)$$

که در آن Y متغیر وابسته، X متغیر مستقل را نشان می‌دهند و β_0 بیانگر عرض از مبدا و β_1 بیانگر شیب

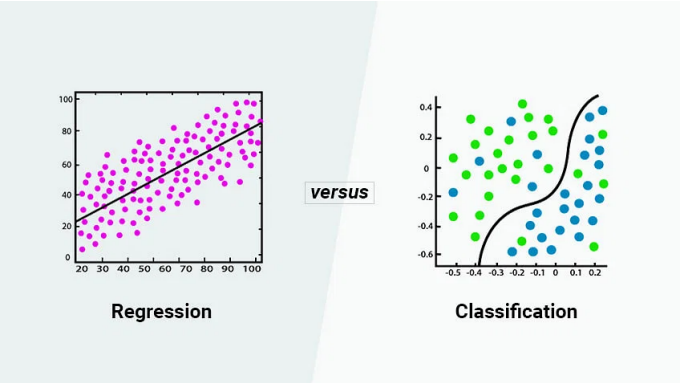
[5] Leonhard Euler
[6] Arthur Cayley
[7] Regression

است و نهایتا ε نمایانگر خطای معادله است که تغییرات غیرقابل توضیح را درنظر می‌گیرد. در مسائل پیچیده‌تری همانند پروژه ما، از رگرسیون غیر خطی استفاده می‌شود که بتواند روابط غیرخطی را نیز مدل کند. فرمول آن به شکل زیر است:

$$Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_p x^p + \varepsilon \qquad (۲-۲)$$

در یادگیری ماشین [۸] رگرسیون روشی برای درک رابطه بین متغیرها یا ویژگی‌های مستقل و یک نتیجه یا متغیر وابسته است. پس از برآورد رابطه بین متغیرهای مستقل و وابسته می‌توان خروجی‌ها را پیش‌بینی کرد. به همین سبب می‌توان گفت رگرسیون یک زمینه مطالعه در آمار است که بخش کلیدی مدل‌های پیش‌بینی در یادگیری ماشین را تشکیل می‌دهد. این روش به عنوان رویکردی برای پیش‌بینی نتایج با مقادیر پیوسته [۹] استفاده می‌شود، بنابراین در پیش‌بینی نتایج حاصل از تحلیل داده‌ها کاربرد دارد.

تفاوت عملکرد رگرسیون دربرابر دسته‌بندی را می‌توان در تصویر زیر مشاهده کرد:



شکل ۲-۲: رگرسیون درمقابل دسته‌بندی

۲-۳-۲ تاریخچه

ریشه‌های تحلیل رگرسیون را می‌توان در اوایل قرن نوزدهم جست‌وجو کرد؛ زمانی که ریاضی‌دان فرانسوی آدرین ماری لژاندر [۱۰]، روش حداقل مربعات [۱۱] را معرفی کرد. با این حال، اصطلاح "رگرسیون" در اواخر قرن نوزدهم، توسط سر فرانسیس گالتون [۱۲] زمانی که او درمورد قد والدین و فرزندان آن‌ها تحقیق می‌کرد، رایج شد. او مشاهده کرد که ارزش‌های مفرط [۱۳] در نسل‌های بعدی به سمت میانگین تمایل به «پسرفت

[8] Machine Learning
[9] Continuous Value
[10] Adrien-Marie Legendre
[11] Mean Square
[12] Sir Francis Galton
[13] Extreme Values

[۱۴]» داشتند و اصطلاح «رگرسیون به میانگین» را ابداع کرد.

۲-۴ شبکه عصبی

۲-۴-۱ تعریف

شبکه‌های عصبی که اغلب به عنوان شبکه‌های عصبی مصنوعی(ANN) [۱۵] شناخته می‌شوند، مدل‌های محاسباتی هستند که از ساختار عصبی مغز انسان الهام گرفته شده‌اند. شبکه‌های عصبی محبوبیت زیادی به دست آورده‌اند و به یک ابزار اساسی در زمینه هوش مصنوعی و یادگیری ماشین تبدیل شده‌اند.

۲-۴-۲ تاریخچه

مفهوم شبکه‌های عصبی به دهه‌های ۱۹۴۰ و ۱۹۵۰ برمی‌گردد، با پیشگامان اولیه مانند وارن مک‌کالوخ[۱۶] و والتر پیتس[۱۷] که مدل‌های ساده شده‌ای از نورون‌ها و منطق دودویی آن‌ها را پیشنهاد دادند. با این حال، تا دهه‌های ۱۹۵۰ و ۱۹۶۰ نظریه شبکه‌های عصبی به مرور توسعه پیدا کرد. ایجاد پرسپترون توسط فرانک روزنبلات[۱۸] در سال ۱۹۵۷ یک قدم مهم بود، زیرا یکی از اولین معماری‌های شبکه‌های عصبی بود که قادر به یادگیری از داده بود.

۲-۴-۳ ساختار

یک شبکه عصبی در هسته خود از لایه‌هایی شامل گره‌های به هم پیوسته تشکیل شده است، که معمولاً به عنوان نورون[۱۹] یا نورون‌های مصنوعی شناخته می‌شوند. این نورون‌ها رفتار نورون‌های زیستی را شبیه‌سازی می‌کنند و اطلاعات را پردازش کرده و انتقال می‌دهند. اجزای کلیدی عبارتند از:

- لایه ورودی: نورون‌ها در لایه ورودی داده‌ها یا ویژگی‌های خام را دریافت کرده و به لایه‌های بعدی منتقل می‌کنند.
- لایه‌های پنهان: یک یا چند لایه پنهان که بین لایه‌های ورودی و خروجی قرار گرفته اند، محاسبات پیچیده‌تری را روی داده‌های ورودی انجام می‌دهند. تعداد لایه‌ها و تعداد نورون‌های پنهان در هر لایه یک انتخاب مهم طراحی شبکه است.

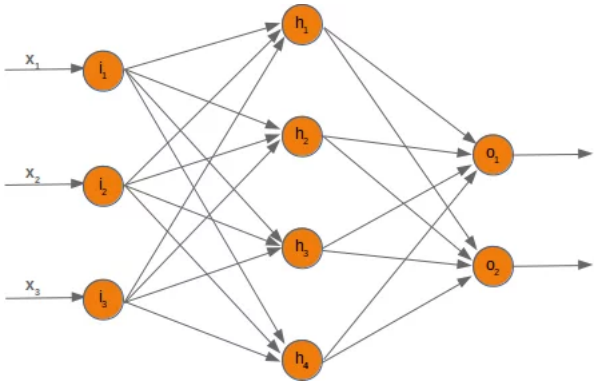
[14] regress
[15] Artificial Neural Networks
[16] Warren McCulloch
[17] Walter Pitts
[18] Frank Rosenblatt
[19] Neuron

- لایه خروجی: لایه خروجی نتایج نهایی را تولید می‌کند که به مسئله مورد نظر بستگی دارد. به عنوان مثال، در یک کار دسته‌بندی[۲۰]، ممکن است احتمالات کلاس‌ها را خروجی دهد، در حالی که در رگرسیون، مقادیر پیوسته تولید می‌کند.

هر نورون در یک شبکه عصبی معمولاً از یک تابع فعال‌سازی استفاده می‌کند که رفتار غیرخطی را نیز به مدل اضافه می‌کند. توابع فعال‌سازی متداول شامل سیگموئید[۲۱]، تانژانت هایپربولیک[۲۲] و واحد یکسو شده‌ی خطی [۲۳] هستند. توابع فعال‌سازی، خروجی نورون را بر اساس ورودی وزن‌دار آن تعیین می‌کنند. فرمول زیر بیانگر یک نورون است:

$$y = f(w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n + b) \qquad (۳-۲)$$

در حالی که x مقادیر ورودی، w وزن‌های مرتبط با هر ورودی، b سوگیری و f تابع فعال‌ساز است و حاصل عبارت برابر با خروجی نورون است.



شکل ۲-۳: یک نمونه از ساختار شبکه عصبی

۲-۴-۴ یادگیری

شبکه‌های عصبی از طریق فرآیندی به نام آموزش از داده‌ها یاد می‌گیرند. متداول‌ترین الگوریتم آموزشی پس‌انتشار[۲۴] است که پارامترهای داخلی شبکه (وزن‌ها[۲۵] و سوگیری‌ها[۲۶]) را به گونه‌ای تنظیم می‌کند که اختلاف بین پیش‌بینی‌ها و مقادیر هدف واقعی(در مجموعه داده آموزشی) به حداقل برسد. برای

[20] Classification
[21] Sigmoid
[22] tanh
[23] ReLU
[24] Back Propagation
[25] Weight
[26] Bias

پس‌انتشار و بهینه‌سازی این پارامترها اغلب از روش کاهش گرادیان[۲۷] استفاده می‌شود. فرمول کاهش گرادیان به شکل زیر است:

$$\theta = \theta - \alpha \cdot \nabla J(\theta) \qquad (۴-۲)$$

در اینجا، θ نمایانگر پارامترهای مدل است. α نرخ یادگیری ($learningrate$) است، که تعیین می‌کند چقدر از هر مرحله در جهت بهینه‌سازی جلو برود. $\nabla J(\theta)$ نمایانگر گرادیان تابع هزینه ($J(\theta)$) نسبت به پارامترهای مدل (θ) است. این گرادیان نشان می‌دهد چگونه تغییرات در پارامترهای مدل باید انجام شود تا بهینه‌سازی انجام شود.

۲-۴-۵ شبکه‌های عصبی عمیق

شبکه‌های عصبی عمیق (Deep Neural Networks یا DNNs) انواع خاصی از مدل‌های یادگیری ماشین هستند که شامل چندین لایه (به عنوان لایه‌های پنهان) از نورون‌ها هستند که به طور سلسله مراتبی و با ساختاری ژرف اطلاعات را از داده‌های ورودی استخراج می‌کنند. این شبکه‌ها با الهام گرفتن از ساختار شبکه‌های عصبی مغز انسان برای مدل کردن الگوها و ویژگی‌های پیچیده در داده‌ها بهره می‌برند.

شبکه‌های عصبی عمیق شامل چندین لایه هستند که هر لایه شامل تعداد زیادی نورون است. اطلاعات از لایه ورودی به لایه‌های پنهان منتقل می‌شوند، و هر لایه پنهان تبدیل‌های مختلفی از داده‌ها را انجام می‌دهد. نتیجه این تبدیل‌ها به لایه‌های بعدی انتقال داده می‌شود تا در لایه‌های آخر به خروجی مورد نظر برسیم.

شبکه‌های عصبی عمیق توانایی یادگیری و تشخیص الگوهای پیچیده در داده‌ها را دارند و به عنوان یکی از قدرتمندترین ابزارهای یادگیری ماشین در حال حاضر شناخته می‌شوند. آن‌ها در بسیاری از حوزه‌های مختلف مانند بینایی ماشین، پردازش زبان طبیعی، ترجمه ماشینی، تشخیص گفتار، بازی‌های ویدئویی، تشخیص الگوها، پیش‌بینی معاملات مالی، و بسیاری دیگر از کاربردها مؤثر هستند.

در عصر اطلاعاتی کنونی، شبکه‌های عصبی عمیق به ویژگی‌ها و نوآوری‌های جدیدی دست یافته‌اند و همچنان در حال توسعه و پیشرفت هستند. از جمله معروف‌ترین معماری‌های شبکه‌های عصبی عمیق می‌توان به شبکه‌های عصبی کانولوشنی (CNNs) برای بینایی ماشین و شبکه‌های عصبی بازگشتی (RNNs) برای پردازش زبان طبیعی اشاره کرد.

[27] Gradient Descent

۲-۵ شبکه عصبی گرافی

۲-۵-۱ تعریف

گراف‌ها، به عنوان ساختارهای داده پیچیده، برای مدل‌سازی روابط و ساختارهای پیچیده در برنامه‌های مختلف دنیای واقعی، مانند شبکه‌های اجتماعی، سیستم‌های توصیه، زیست‌شناسی و شبکه‌های حمل‌ونقل استفاده می‌شوند. تجزیه و تحلیل و استخراج اطلاعات معنی دار از چنین داده‌های گرافی یک مشکل اساسی در یادگیری ماشین و هوش مصنوعی است. شبکه‌های عصبی گرافی[۲۸] (GNN) به عنوان یک رویکرد قدرتمند برای مدل‌سازی و تحلیل داده‌های ساختار یافته گراف در حوزه‌های مختلف ظهور کرده‌اند [۱۸] . با افزایش دسترسی به داده‌های شبکه در زمینه‌هایی مانند شبکه‌های اجتماعی، زیست‌شناسی، سیستم‌های توصیه [۲۹] و حمل و نقل، نیاز روزافزونی به روش‌های موثر برای استخراج اطلاعات معنادار از این ساختارهای پیچیده وجود دارد. GNNها با استفاده از اتصالات و اطلاعات رابطه‌ای موجود در گراف‌ها، راه حل مناسبی را ارائه می‌دهند تا دانش‌هایی را از ساختار دادگان استخراج کرده و بیاموزند. در ادامه، توضیحی در خصوص این شبکه‌ها، ضرورت و مزایایشان، توسعه تاریخی آن‌ها و کاربردهایشان ارائه می‌دهیم.

۲-۵-۲ تاریخچه

پیداییش مفهوم GNNها را می‌توان در اواخر دهه ۱۹۹۰ جست‌وجو کرد، زمانی که اسپردوتی [۳۰] و همکاران [۱۱] برای اولین بار از شبکه‌های عصبی برای گراف‌های غیر چرخه‌ای جهت دار استفاده کردند. این کار اولیه پایه و اساس مطالعات بعدی بر روی GNNها را ایجاد کرد و انگیزه تحقیقات بیشتر در این زمینه را فراهم کرد. از آن زمان، GNNها به طور قابل توجهی تکامل یافته‌اند و ایده‌هایی از پردازش سیگنال گراف، شبکه‌های عصبی کانولوشنال و یادگیری عمیق را در خود جای داده‌اند. با این حال، در دهه‌های ۲۰۰۰ و ۲۰۱۰ بود که GNNها به دلیل پیشرفت در یادگیری عمیق و نیاز به مدیریت ساختارداده‌های نامنظم مانند گراف‌ها، توجه زیادی را به خود جلب کردند.

۲-۵-۳ ساختار

در مراحل اولیه نظریه گراف، محققان الگوریتم‌های گراف مختلفی را برای کارهایی مانند خوشه‌بندی [۳۱]، تجزیه و تحلیل مرکزیت و تشخیص جامعه توسعه دادند. این الگوریتم‌ها پایه و اساس پیشرفت‌های بعدی در GNNها را تشکیل دادند. تا جایی که امروزه، مفهوم شبکه‌های عصبی گراف (GNN) به عنوان

[28] Graph Neural Networks
[29] Recommender Systems
[30] Sperduti
[31] Clustering

یک رویکرد قدرتمند در زمینه تحلیل گراف‌ها و داده‌های ساختاریافته مطرح شده است. در مراحل اولیه نظریه گراف، الگوریتم‌ها معمولاً بر اساس مفاهیمی مانند اتصالات گره‌ها، فاصله‌ها، ویژگی‌های گره‌ها و ساختار گراف عمل می‌کردند. با گذشت زمان، ترکیب مفاهیم نظریه گراف با تکنولوژی‌های یادگیری عمیق به وجود آمد و این ترکیب نه تنها امکان استفاده از اطلاعات ساختار گراف را فراهم آورد بلکه امکان بهبود الگوریتم‌ها و روش‌ها با استفاده از توانایی‌های شبکه‌های عصبی را ایجاد کرد. این ترکیب ابتدایی‌ترین و پایه‌ای‌ترین نسخه از GNNها را شکل داد و تا به امروز به یکی از پرکاربردترین مدل‌های یادگیری ماشین در زمینه تحلیل گراف‌ها تبدیل شده است.

۱. نمایش گراف: ورودی GNNها یک گراف است که معمولاً به صورت یک جفت ماتریس نشان داده می‌شود: یک ماتریس مجاورت (A) و یک ماتریس ویژگی گره (X).

- ماتریس مجاورت (**A**): این ماتریس ارتباط بین گره‌ها را در گراف توصیف می‌کند. اغلب دودویی است، در صورتی که یک یال بین گره i و گره j وجود داشته باشد، A[i, j] = 1 و در غیر این صورت ۰ است. با این حال، می‌توان به آن برای نمایش نقاط قوت لبه در یک گراف وزن‌دار نیز وزن اختصاص داد.
- ماتریس ویژگی گره (**X**): این ماتریس شامل بردارهای ویژگی برای هر گره در گراف است. این ویژگی‌ها می‌توانند ویژگی‌های گره مانند متن، مقادیر عددی یا جاسازی‌ها [۳۲] را نشان دهند.

۲. مقداردهی اولیه: شبکه‌های عصبی گرافی با مقداردهی اولیه جاسازی گره‌ها شروع می‌شوند. این جاسازی‌ها را می‌توان به‌صورت تصادفی یا با استفاده از جاسازی‌های از پیش آموزش‌دیده مقداردهی کرد.

۳. ارسال پیام: عملیات اصلی در یک شبکه عصبی گرافی ارسال پیام است. ارسال پیام شامل انتقال اطلاعات بین گره‌های همسایه در گراف است. این مرحله را می‌توان چندین بار تکرار کرد و به گره‌ها اجازه می‌دهد اطلاعات را از همسایگان دورتر جمع‌آوری کنند.

- عملکرد پیام: این تابع نحوه جمع‌آوری اطلاعات از گره‌های همسایه را مشخص می‌کند. انتخاب‌های متداول شامل جمع وزنی، پیچش‌های گراف[۳۳] یا مکانیسم‌های توجه[۳۴] است.
- عملکرد تجمیع: اطلاعات جمع‌آوری شده با جاسازی فعلی گره ترکیب می‌شود. توابع تجمیع متداول عبارتند از الحاق، افزودن بر حسب عنصر، یا تجمعات خاص گراف.

[32] Embeddings

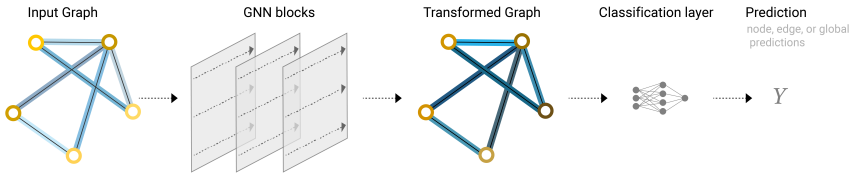
[33] Graph Convolutions

[34] Attention Mechanisms

۴. جمع‌آوری و بروزرسانی: پس از تجمیع اطلاعات از همسایگان، هر گره مقدار جاسازهای خود را به‌روز می‌کند. این مرحله جاسازی قبلی گره را با اطلاعات جمع‌آوری شده از مرحله ۳ ترکیب می‌کند.

۵. بازخوانی[۳۵]-ادغام[۳۶]: برای بدست آوردن یک نمایش در سطح گراف (در صورت نیاز)، می‌توان عملیات بازخوانی یا ادغام را انجام داد. این مرحله اطلاعات را از تمام گره‌ها در یک نمایش در سطح گراف جمع می‌کند.

۶. خروجی: جاسازهای نهایی گره (یا نمایش در سطح گراف) را می‌توان برای وظایف مختلف، مانند دسته‌بندی گره، دسته‌بندی گراف، پیش‌بینی پیوند، یا رگرسیون استفاده کرد.



شکل ۲-۴: یک نمونه از عملکرد شبکه عصبی گرافی در وظیفه دسته‌بندی

شبکه‌های عصبی گراف دارای معماری‌ها و گونه‌های مختلفی هستند، از جمله شبکه‌های پیچشی گراف (GCN)، شبکه‌های توجه گراف (GAT)، GraphSAGE. هر یک از این معماری‌ها ممکن است روش‌های متفاوتی برای انجام ارسال و تجمیع پیام داشته باشند، اما ساختار اصلی که در بالا توضیح داده‌شد، مشابه باقی می‌ماند.

۲-۵-۴ ضرورت

یادگیری عمیق به ابزار مهمی در تحلیل داده دنیای امروز بدل شده است اما ضعف‌هایی دارد که برخی از آن‌ها را شبکه عصبی گرافی به خوبی برطرف کرده است. در حالی که یادگیری عمیق به شکل قوی الگوهای پنهان داده‌های اقلیدسی را پیدا و ثبت می‌کند؛ تعداد فزاینده‌ای از کاربردها وجود دارد که در آن‌ها داده‌ها به شکل گراف نمایش داده می‌شوند و برای شبکه‌های عمیق معمول به خوبی قابل درک نیستند. باتوجه به این کاربردهای رو به افزایش و مهم، تحقیقات و توسعه بر روی شبکه‌های عصبی گرافی از ضرورت‌ها دنیای امروز در عرصه هوش مصنوعی است. بنابراین، یکی از مزیت‌های کلیدی (GNN)ها توانایی آن‌ها برای مدیریت داده‌های غیر اقلیدسی است که در مجموعه داده‌های ساختاریافته گراف رایج است. شبکه‌های عصبی سنتی برای داده‌های شبکه مانند تصاویر که در آن هر نقطه داده مستقل از همسایگان خود است، طراحی شده‌اند. در مقابل، شبکه‌های عصبی گرافی می توانند روابط بین گره‌ها را در یک گراف ثبت کنند، که امکان مدل سازی دقیق‌تر تعاملات و وابستگی‌های پیچیده را فراهم می‌کند.

[35] Readout
[36] Pooling

این باعث می‌شود که GNNها برای کارهایی مانند دسته‌بندی گره، دسته‌بندی گراف، پیش‌بینی پیوند و تشخیص ناهنجاری مناسب باشنند [۱۳، ۲۰].

به عنوان مثال، یک سیستم یادگیری مبتنی بر گراف می‌تواند با یادگیری از تعاملات بین کاربران و محصولات و دانش استخراج شده، برای ارائه پیشنهادات بسیار دقیق استفاده کند. در شیمی، مولکول‌ها به صورت گراف مدل‌سازی می‌شوند و فعالیت زیستی آن‌ها برای کشف دارو باید شناسایی شود. در یک شبکه استنادی، مقالات از طریق استناد به یکدیگر مرتبط می‌شوند و باید در گروه‌های مختلف دسته‌بندی شوند که این امر با به‌کارگیری شبکه‌های عصبی گرافی تسهیل می‌شود. در بسیاری از حوزه‌ها مانند علوم اجتماعی، حمل و نقل، زیست‌شناسی و شبکه‌های اجتماعی، نیاز به مدل‌های رگرسیون بر روی گراف‌ها وجود دارد. این مدل‌ها می‌توانند به ما کمک کنند تا مقادیر پیوسته‌ی مهم و معناداری را در هر گره در این حوزه‌ها پیش‌بینی کنیم. با توجه به پیچیدگی و ارتباطات پیچیده درون گراف‌ها، استفاده از شبکه‌های عصبی گرافی می‌تواند به ما کمک کند تا به نتایج بهتر و دقیق‌تری نسبت به بسیاری از مدل‌های هوش مصنوعی دیگر برای رگرسیون دست یابیم. [۱۸]

موفقیت GNNها را می توان به توانایی آن‌ها در جمع آوری اطلاعات از گره‌های همسایه و انتشار آن از طریق گراف نسبت داد. این فرآیند، که به عنوان ارسال پیام[۳۷] شناخته می‌شود، به GNNها اجازه می‌دهد تا هم اطلاعات محلی و هم اطلاعات سراسری را ضبط کنند، و آن‌ها را قادر می‌سازد تا بازنمایی‌های غنی از اطلاعات را یاد بگیرند که ویژگی‌های ساختاری و معنایی گراف را به تصویر می‌کشد.

۲-۵-۵ کاربردها

جهت بررسی کاربردهای موجود در حوزه شبکه‌های عصبی گرافی، به دسته‌بندی کلی وظایف مورد بررسی این زمینه می‌پردازیم و برای هرکدام از این وظایف، کاربردها و صنایعی مثال می‌زنیم. در اینجا برخی از وظایف رایج که GNNها برای آن‌ها استفاده می‌شوند آورده شده است:

- دسته‌بندی[۳۸] گره: در این وظیفه، از GNNها برای دسته‌بندی گره‌ها(راس‌ها) در یک گراف به دسته‌ها یا برچسب‌های از پیش تعریف شده استفاده می‌شود. به عنوان مثال، دسته‌بندی کاربران در یک شبکه اجتماعی به گروه‌های مختلف یا شناسایی گره‌های مخرب در یک شبکه کامپیوتری.
- دسته‌بندی گراف: به جای دسته‌بندی گره‌های جداگانه، GNNها می‌توانند کل گراف‌ها را دسته‌بندی کنند. به عنوان مثال، دسته‌بندی گراف‌های مولکولی به عنوان فعال یا غیرفعال برای یک کار خاص در کشف دارو.
- پیش‌بینی پیوند[۳۹]: GNNها می‌توانند لبه‌های گمشده یا بالقوه (اتصالات) بین گره‌ها را در یک گراف پیش‌بینی کنند. این قابلیت در سیستم‌های توصیه، تجزیه و تحلیل شبکه‌های اجتماعی و

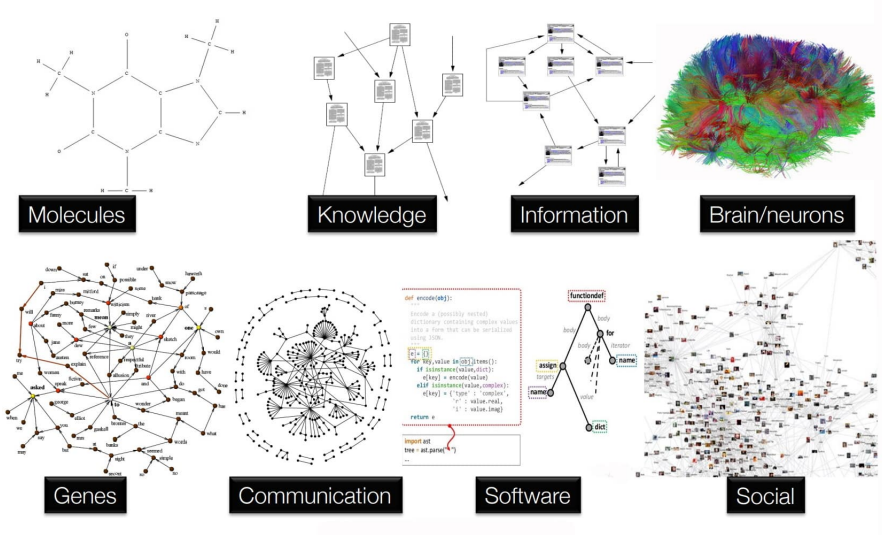
[37] Message Passing
[38] Classification
[39] Link Prediction

تکمیل گراف دانش ارزشمند است.

- تولید گراف[۴۰]: از GNNها می‌توان برای تولید گراف‌های جدیدی استفاده کرد که ویژگی‌های ساختاری مشابهی را با مجموعه معینی از گراف‌های ورودی نشان می‌دهند. این ابزار در تولید ساختارهای مولکولی یا شبیه‌سازی شبکه‌های اجتماعی واقعی مفید است.
- تشخیص جامعه[۴۱]: GNNها می‌توانند جوامع یا خوشه‌هایی[۴۲] از گره‌ها را در یک گراف شناسایی کنند و ساختار جامعه نهفته در دادگان را آشکار کنند. این قابلیت در تجزیه و تحلیل شبکه‌های اجتماعی، تجزیه و تحلیل شبکه‌های استنادی[۴۳] و موارد دیگر اعمال می‌شود.
- تشخیص ناهنجاری: تشخیص گره‌ها یا زیرگراف‌های غیرعادی در یک گراف برای کاربردهای مختلف مانند تشخیص تقلب، امنیت شبکه و کنترل کیفیت ضروری است.
- سیستم‌های توصیه: از GNNها می‌توان برای ساخت سیستم‌های توصیه‌ای استفاده کرد که بر اساس تعاملات یا ترجیحات آن‌ها در یک گراف، توصیه‌های شخصی‌سازی‌شده را به کاربران ارائه می‌دهند.
- جاسازی گراف: GNNها جاسازی‌ها (نمایش‌های برداری) را برای گره‌ها یاد می‌گیرند که می‌توانند در کارهای یادگیری ماشین استفاده شوند. این جاسازی‌ها اطلاعات ساختاری و معنایی گره‌ها در گراف را درخود جای می‌دهند.
- رگرسیون[۴۴] گره: به جای دسته‌بندی، GNNها می‌توانند وظایف رگرسیونی را انجام دهند و یک مقدار پیوسته مرتبط با هر گره را پیش‌بینی کنند. به عنوان مثال، پیش‌بینی قیمت یک خانه بر اساس ویژگی‌ها و ارتباط محله‌ای آن.
- رگرسیون گراف: مشابه رگرسیون گره، GNNها می‌توانند مقادیر پیوسته مرتبط با کل گراف‌ها را پیش‌بینی کنند. این می‌تواند در کاربردهایی مانند پیش‌بینی خواص مولکول‌ها بر اساس گراف‌های مولکولی آن‌ها استفاده شود.

این وظایف، تطبیق‌پذیری GNNها را در مدیریت انواع داده‌های ساختاریافته گرافی و کاربردهای آن‌ها در حوزه‌های مختلف از جمله شبکه‌های اجتماعی، زیست‌شناسی، سیستم‌های توصیه و غیره نشان می‌دهد. انتخاب مدل شبکه‌عصبی و تعیین وظیفه متناظر با مسئله به صورت مسئله خاص و مجموعه داده مورد بررسی بستگی دارد.

[40] Graph Generation
[41] Community Detection
[42] Cluster
[43] Citation Networks
[44] Regression



شکل ۲-۵: کاربردهای شبکه عصبی گرافی

۲-۵-۶ رگرسیون گره‌ها در شبکه‌های عصبی گرافی

رگرسیون گره در شبکه‌های عصبی گرافی به وظیفه پیش‌بینی یک مقدار پیوسته مرتبط با هر گره در یک گراف اشاره دارد. در این زمینه، گره‌ها می‌توانند موجودیت‌ها یا نقاط داده مختلفی را نشان دهند و هدف، پیش‌بینی یک مقدار عددی (به عنوان مثال، یک عدد حقیقی) برای هر یک از این گره‌ها است. بنابراین، این شبکه‌ها نیز ساختاری همانند آنچه در بخش ۲-۵-۳ گفته شد دارند. با این خصوصیت که مقادیر قابل پیش‌بینی به صورت مقادیر پیوسته هستند.

رگرسیون گره با استفاده از GNNها در زمینه‌های مختلفی مانند سیستم‌های توصیه، زیست‌شناسی(پیش‌بینی خواص پروتئین)، حمل و نقل(پیش‌بینی جریان ترافیک برای بخش‌های مختلف جاده) و امور مالی(پیش‌بینی قیمت دارایی) کاربرد دارد.

انتخاب معماری GNN و هایپرپارامترها می‌تواند به طور قابل توجهی بر عملکرد وظایف رگرسیون گره تأثیر بگذارد و انتخاب مدل مناسب و تنظیم دقیق آن برای مجموعه داده خاص بسیار مهم است.

فصل سوم

کارهای پیشین

۳-۱ مقدمه

رگرسیون گره‌های گراف یک کار پایه در شبکه‌های عصبی گرافی است که هدف آن پیش‌بینی مقادیر پیوسته مرتبط با هریک از گره‌های گراف است. در طول سال‌ها، محققان پیشرفت قابل توجهی در توسعه روش‌ها و برنامه‌های کاربردی برای رگرسیون گره گراف داشته‌اند. این فصل مروری بر فعالیت‌های قبلی در این زمینه ارائه می‌کند و مقالات تحقیقاتی کلیدی و مشارکت‌های آن‌ها را بیان می‌کند. اگرچه ممکن است تمرکز اصلی برخی از این روش‌ها بر وظایفی غیر از رگرسیون بوده باشد، در هدف رگرسیون هم کاربردی بودند و تاثیر به‌سزایی داشته‌اند.

۳-۲ روش‌های سنتی

در این بخش توضیحاتی برای هر یک از الگوریتم‌ها و رویکردهای سنتی مورد استفاده برای رگرسیون در گراف‌ها قبل از ظهور شبکه‌های عصبی گرافی ارائه می‌دهیم.

۳-۲-۱ رگرسیون خطی روی گراف‌ها

- رویکرد: مدل‌های رگرسیون خطی[1] در مسائلی به گراف‌ها تعمیم می‌یابند که در آن مقدار هدف هر گره به عنوان ترکیبی خطی از ویژگی‌های خود و گره‌های همسایه‌اش پیش‌بینی می‌شود. این رویکرد یک رابطه ساده و خطی بین مقادیر گره را فرض می‌کند. [۵]
- روش: اساساً حل یک سیستم معادلات خطی است که در آن ضرایب،ط نشان دهنده قدرت تأثیر گره‌های همسایه است.
- کاربردها: رگرسیون خطی بر روی گراف‌ها در سناریوهایی اعمال شد که در آن رابطه بین گره‌ها و ویژگی‌های آنها تقریباً خطی است، مانند شبکه‌های اجتماعی برای پیش‌بینی رفتارهای کاربر.

۳-۲-۲ تکنیک‌های منظم‌سازی گراف

- رویکرد: روش منظم‌سازی گراف[2]، مدل‌های رگرسیون را با افزودن یک مولفه جریمه بر اساس ساختار گراف، ارتقا می‌دهد. این کار موجب برآورد پیش‌بینی‌های دقیق‌تر و هموارترمی‌شود. [۲۱]
- روش‌شناسی: معمولاً از منظم‌سازی لاپلاسی استفاده می‌شود که شباهت را در پیش‌بینی گره‌های متصل مورد توجه قرار می‌دهد.

[1] Linear Regression

[2] Graph Regularization

- کاربردها: روش‌های منظم‌سازی به‌ویژه زمانی مفید بودند که یک دانش قبلی از گراف نشان می‌داد که گره‌های متصل در یک گراف باید مقادیر پیش‌بینی‌شده مشابهی داشته باشند، به عنوان مثال، در سیستم‌های توصیه.

۳-۲-۳ روش‌های هسته

- رویکرد: گراف‌ها از طریق هسته گراف[۳] به بردارهای ویژگی تبدیل می‌شوند. سپس از این بردارها به عنوان ورودی برای مدل‌های رگرسیون سنتی استفاده می‌شود. [۱۷]
- روش‌شناسی: هسته‌های گراف، شباهت گراف را کمّی(عددی) می‌کنند. این هسته‌ها شامل تکنیک‌های مختلفی مانند هسته‌های پیاده‌روی تصادفی، هسته‌های انتشار و هسته‌های کوتاه‌ترین مسیر هستند.
- کاربردها: روش‌های هسته زمانی مفید هستند که اندازه‌گیری و بدست‌آوردن شباهت گراف ضروری است، که اغلب درش شیمی و زیست‌شناسی یافت می‌شود.

۳-۲-۴ پیاده‌روی تصادفی و انتشار گراف

- رویکرد: اطلاعات با استفاده از تکنیک‌های پیاده‌روی تصادفی[۴] یا انتشار گراف[۵] در گراف‌ها منتشر می‌شود. مقادیر به دست آمده تأثیر بین گره‌ها را نشان می‌دهند. [۷]
- روش‌شناسی: پیج رنک شخصی شده و هسته‌های حرارتی نمونه‌هایی از الگوریتم‌های انتشار اطلاعات هستند.
- کاربردها: این روش‌ها زمانی مناسب استفاده هستند که درک روابط بین گره‌ها بر اساس اتصال آنها و انتشار اطلاعات بسیار مهم است، همانطور که در سیستم‌های توصیه یا پیش‌بینی گسترش بیماری دیده می‌شود.

۳-۲-۵ پردازش سیگنال گراف

- رویکرد: با درنظر گرفتن گراف‌ها به عنوان سیگنال، این رویکرد از فیلترها و تبدیل‌های گراف مشابه پردازش سیگنال سنتی، برای تجزیه و تحلیل و دستکاری داده‌های گراف استفاده می‌کند. [۱۵]

[3] Graph kernels
[4] Random Walk
[5] Graph Diffusion

- روش‌شناسی: شامل مفاهیمی از پردازش سیگنال کلاسیک، مانند تبدیل‌های فوریه گراف و بانک‌های فیلتر گراف است.
- کاربردها: این تکنیک‌ها زمانی ارزشمند هستند که نیاز به تجزیه و تحلیل ویژگی‌های سیگنال‌مانند در گراف‌ها است که در سناریوهایی مانند شبکه‌های حسگر و پردازش تصویر روی گراف‌ها یافت می‌شود.

۳-۲-۶ نظریه گراف طیفی

- رویکرد: این رویکرد با استفاده از مقادیر ویژه و بردارهای ویژه ماتریس لاپلاسی گراف، گراف‌ها را به فضاهای با ابعاد پایین‌تر می‌برد. [۲]
- روش‌شناسی: روش‌هایی مانند نقشه‌های ویژه لاپلاسی برای ایجاد جاسازی‌ها برای مدل‌های رگرسیون بعدی استفاده می‌شوند.
- کاربردها: روش‌های طیفی زمانی مفید هستند که بخواهیم ابعاد گراف‌های پیچیده را کاهش دهیم و در عین حال ویژگی‌های ساختاری آن‌ها را حفظ کنیم، مانند تقسیم‌بندی تصویر.

۳-۲-۷ انتشار برچسب

- رویکرد: روش‌های انتشار برچسب، برچسب‌های گره شناخته شده را از طریق گراف منتشر می‌کنند تا برچسب‌ها یا مقادیر گره‌های بدون برچسب را پیش‌بینی کنند. [۲۲]
- روش‌شناسی: الگوریتم‌هایی مانند الگوریتم انتشار برچسب [۶] (LPA) و تکنیک‌های یادگیری نیمه‌نظارتی استفاده می‌شود.
- کاربردها: انتشار برچسب زمانی موثر است که چند نقطه داده برچسب‌گذاری شده(یا مقداردهی شده) داشته باشیم و بخواهیم برای کل گراف برچسب‌ها(یا مقادیر) را پیش‌بینی کنیم، مانند تشخیص جامعه و یادگیری نیمه نظارت شده.

۳-۲-۸ مهندسی ویژگی گراف

- رویکرد: مهندسی ویژگی گراف شامل ایجاد دستی ویژگی‌های اطلاعاتی است که جنبه‌های مختلف ساختار گراف، مانند درجه گره، مرکزیت، و ضرایب خوشه‌بندی را به تصویر می‌کشد. [۱۲]
- روش‌شناسی: دانش دامنه برای طراحی و استخراج ویژگی‌های مرتبط برای وظایف رگرسیون استفاده می‌شود.

[6] Label Propagation Algorithm

- کاربردها: این رویکرد زمانی اعمال می‌شود که اطلاعات خاص دامنه در مورد ساختار گراف در دسترس باشد و بتوان آن را به ویژگی‌های مهندسی شده ترجمه کرد. مانند تجزیه و تحلیل شبکه‌های اجتماعی و شبکه‌های ارجاعات و اسناد.

۳-۲-۹ استخراج زیرگراف

- رویکرد: هدف روش‌های استخراج زیرگراف کشف و تجزیه و تحلیل زیرگراف‌هایی در گراف‌های بزرگ‌تر است که مقادیر هدف قابل پیش‌بینی دارند. [۳]
- روش‌شناسی: الگوریتم‌های استخراج زیرگراف مکرر معمولاً برای شناسایی زیرساخت‌های تکرارشونده استفاده می‌شوند.
- کاربردها: زمانی که شناسایی الگوهای محلی موثر بر متغیر هدف در گراف نیاز باشد، استخراج زیرگراف ارزشمند است، مانند بیوانفورماتیک و تشخیص ناهنجاری شبکه.

۳-۳ خلاصه

این رویکردهای سنتی ابزارهای ارزشمندی را برای مدل‌سازی پیش‌بینی‌کننده در داده‌های ساختاریافته گرافی ارائه می‌کنند. با این حال، آن‌ها اغلب بر فرضیات ساده‌سازی تکیه می‌کردند یا به مهندسی ویژگی‌های دستی گسترده نیاز داشتند. ظهور شبکه‌های عصبی گرافی با اجازه دادن به مدل‌ها برای یادگیری خودکار روابط و ویژگی‌های پیچیده از داده‌های گراف، انقلابی در این زمینه ایجاد کرد و آن‌ها را به انتخاب اول برای بسیاری از وظایف رگرسیون مبتنی بر گراف تبدیل کرد.

فصل چهارم

دادگان، آماده‌سازی و مراحل مقدماتی

۴-۱ مقدمه

در حوزه تحقیقات مبتنی بر داده، نمی‌توان اهمیت مجموعه داده‌های با ساختار و کیفیت بالا را اغراق کرد. داده‌ها، در حالت خام خود، اغلب دارای معانی نهفته ارزشمندی هستند، با این حال محقق موظف است که این پتانسیل را از طریق پیش‌پردازش دقیق داده‌ها قابل دسترس کند. در این فصل، ما به بررسی خود داده‌ها می‌پردازیم.

این فصل به ارائه یک گزارش جامع از مجموعه داده‌های ما، گام‌های برداشته شده برای پیش‌پردازش آن‌ها و آماده‌سازی اولیه جهت استفاده کامل از پتانسیل تحلیلی داده‌ها اختصاص دارد. از آغاز فرآیند پروژه، انتخاب و آماده‌سازی مجموعه داده‌ها یک مسئله‌ی محوری بوده است. چراکه داده‌ها بستری را تشکیل می‌دهند که تحلیل‌های ما بر اساس آن شکل می‌گیرند. به این ترتیب، اهتمام و دقت اعمال شده برای مدیریت مجموعه داده‌های ما مستقیماً بر کیفیت و اعتبار یافته‌های تحقیق ما تأثیر می‌گذارد.

این فصل، با کاوش در منشأ، ساختار و معیارهای انتخاب دادها آغاز می‌شود. در ادامه، تمرکز را بر مرحله پیش‌پردازش داده‌ها معطوف می‌کنیم. پیش‌پردازش شامل رویه‌های ضروری مختلفی مانند تمیز کردن، تبدیل و مهندسی ویژگی است. در طول این مرحله است که ما تلاش می‌کنیم تا داده‌های خام خود را به حالتی تبدیل کنیم که قابل استفاده برای تکنیک‌های تحلیلی پیشرفته شوند. منطق و روش ما در پشت هر مرحله پیش‌پردازش به تفصیل گزارش شده‌است. در نهایت، اقدامات نهایی انجام شده برروی دادگان (جهت اجرای مراحل بعدی تجزیه و تحلیل داده‌ها) را توضیح می‌دهیم. این فصل به تشریح ابزارها، نرم‌افزارها و محیط‌های مورد استفاده در تحقیق ما می‌پردازد. موفقیت فعالیت‌های تحلیلی ما وابسته به این انتخاب‌های پایه‌ای است. مباحث این فصل نه تنها جنبه‌های مرتبط با مجموعه دادگان را روشن می‌کند، بلکه پایه محکمی برای فصل‌های بعدی ایجاد می‌کند. از طریق انتخاب عاقلانه، پیش‌پردازش مناسب، و آماده‌سازی مجموعه داده‌های خود، پایه‌ی قدم‌های بعدی پروژه را مستحکم می‌کنیم.

۴-۲ انتخاب دادگان

همانطور که در ۴-۱ گفته شد، داده‌ها بستری را تشکیل می‌دهند که تحلیل‌های ما بر اساس آن شکل می‌گیرند. بنابراین انتخاب آن‌ها از کارهای اساسی تجزیه‌وتحلیل اطلاعات است و باید با دقت و مطالعه کافی صورت گیرد. برای این منظور مجموعه دادگان مورد انتخاب باید ویژگی‌هایی داشته باشد که آن را به گزینه‌ی مناسبی برای وظیفه مورد بررسی ما تبدیل کند.

۴-۲-۱ ساختار شبکه‌ای

یکی از امور حیاتی در انتخاب دادگان، ساختار شبکه‌ای آن‌هاست. زیرا هدف ما تحلیل و بررسی عملکرد مدل‌های شبکه‌عصبی بر روی ساختار داده گرافی است و ما در گراف، تحلیل‌های خود را بر مبنای ارتباطات گره‌ها و ویژگی‌های موجود در شبکه انجام می‌دهیم(بر خلاف آنچه که در تحلیل داده‌های اقلیدسی انجام

می‌شود). از این رو، دادگان انتخابی باید هم از نظر موضوعی مناسب باشند وهم ساختاری شبکه‌ای داشته باشند که گره‌ها، ارتباطات بین آن‌ها و ویژگی‌های متناظرشان را به خوبی نشان دهد.

۴-۲-۲ مقادیر هدف پیوسته گره‌ها

در حوزه رگرسیون گره‌ها با استفاده از شبکه‌های عصبی گرافی، مقادیر هدف مورد پیش‌بینی معمولاً مقادیر پیوسته هستند. درحالی که این مقادیر با توجه به وظیفه مورد بررسی می‌تواند کاملا متفاوت باشد(برای مثال، در وظیفه دسته‌بندی، ما به برچسب‌هایی با مقادیر ثابت و محدود نیاز داریم و یا در پیش‌بینی ارتباطات، هدف ما پیش‌بینی وجود یا عدم وجود یال‌هاست). بنابراین، مجموعه دادگان باید حاوی مقادیر هدف پیوسته برای گره‌ها باشد تا بتوانیم از مدل‌های رگرسیون استفاده کنیم.

۴-۲-۳ ابعاد داده

یکی از موارد مهم در انتخاب دادگان مناسب برای تحقیقات رگرسیون گره‌ها با شبکه‌های عصبی گرافی، ابعاد داده است. ابعاد داده به اندازه ویژگی‌های کلی داده‌ها اشاره دارد که از طریق داده‌ها قابل مشاهده و قابل اندازه‌گیری هستند. در داده‌ساختار گرافی مواردی از جمله، تعداد گره‌ها، تعداد یال‌ها، تعداد ویژگی‌ها، درجه گره‌ها و تراکم گراف از اهمیت بالایی برخوردارند و دادگان انتخابی باید از ابعاد کافی و متناسب برخوردار باشد تا بتوان تحلیلی معتبر روی آن‌ها انجام داد.

۴-۲-۴ معتبر بودن

اطمینان از معتبر بودن دادگان بسیار مهم است چراکه تاثیر اساسی در عملکرد الگوریتم‌ها و نتایج و تحلیل‌های ما می‌گذارد. برای اطمینان از این موضوع، دادگان باید از منابع معتبر یا تولیدشده با روش‌های استاندارد به دست آمده باشند. همچنین، پیش از استفاده از دادگان، آن‌ها باید مورد پیش‌پردازش و تصفیه دقیق قرار گیرند.

۴-۲-۵ بکر بودن

در برخی کاربردها و به طور خاص پروژه‌ی ما، بکر بودن دادگان حائز اهمیت است. چرا که برای اکثر دادگان نامدار در حوزه گراف مقالات و فعلالیت‌های متعدد ثبت و ارائه شده‌است. اما با انتخاب یک مجموعه داده‌ی بکر می‌توان فعالیت‌ها و نتایج بدیع و ارزشمندی ارائه کرد.

۴-۲-۶ Wiki-Squirrel

امروزه مجموعه دادگان متعددی در حوزه تحلیل گراف جمع‌آوری و ارائه شده‌اند. اما اکثرا فاقد مقادیر پیوسته گره‌ها بوده و برای دیگر وظایف مانند دسته‌بندی مناسب هستند و پیدا کردن مجموعه دادگان مناسب رگرسیون گره‌ها یکی از چالش‌های اساسی پروژه بود، حال آنکه مجموعه داده انتخابی بکر هم باشد. با این حال می‌توان به مجموعه داده OGB-LSC[۹]، MoleculeNet[۱۹]، LRGB[۶] و ZINC[۱] به عنوان چند مجموعه داده معتبر و پر استفاده اشاره کرد.

ما برای تحقق بخشیدن به اهداف پروژه و درنظر گرفتن ویژگی‌های مطلوب مجموعه‌دادگان از مجموعه دادگان Wiki-Squirrel (Wikipedia Article Networks) [۱۴] استفاده کردیم که ویژگی‌های مد نظر ما را برآورده می‌کند. این مجموعه‌داده توسط دانشگاه استنفورد جمع‌آوری شده که اعتبار بالایی به آن می‌بخشد و شبکه‌هایی شامل گره‌های مبدا و مقصد، یال‌هایی بدون جهت، مقادیر پیوسته گره‌ها همراه با ویژگی‌هایشان را نشان می‌دهد.

داده‌ها از ویکی‌پدیای انگلیسی (دسامبر ۲۰۱۸) جمع‌آوری شده‌اند. این مجموعه‌های داده نمایانگر شبکه‌های صفحه به صفحه در موضوعات خاص (مانند آفتاب‌پرست، تمساح‌ها و سنجاب‌ها) هستند. گره‌ها مقالات را بازنمایی می‌کنند و یال‌ها پیوندهای متقابل بین آنها هستند. فایل‌های CSV مربوط به یال‌ها شامل ارتباطات هستند و گره‌ها از شماره ۰ شماره‌گذاری شده‌اند. فایل‌های JSON مربوط به ویژگی‌های مقالات شامل خصیصه‌های مقالات هستند؛ هر کلید یک شناسه صفحه(گره) است و ویژگی‌های گره به صورت لیست‌هایی از اعداد آمده‌اند. حضور یک ویژگی در لیست ویژگی‌ها به معنی ظاهر شدن یک اسم معنی‌دار در متن مقاله ویکی‌پدیا است. فایل CSV مربوط به مقادیر هدف شامل شناسه گره‌ها و میانگین ترافیک ماهانه بین اکتبر ۲۰۱۷ تا نوامبر ۲۰۱۸ برای هر صفحه است(که یک مقدار پیوسته مناسب برای کاربرد ماست). برای هر شبکه صفحه به صفحه، تعداد گره‌ها و یال‌ها به همراه برخی اطلاعات توصیفی دیگر آمده‌اند. درواقع Wiki-Squirrel خود شامل سه مجموعه داده است. که جزئیاتشان به شکل زیر است:

۱. **Chameleon Dataset**: از ۲۲۷۷ گره همراه با ۳۱۴۲۱ لبه تشکیل شده‌است و مربوط به مقالات مرتبط با آفتاب‌پرست می‌باشد.

۲. **Squirrel Dataset**: از ۵۲۰۱ گره همراه با ۱۹۸۴۹۳ لبه تشکیل شده‌است و مربوط به مقالات مرتبط با سنجاب می‌باشد.

۳. **Corocodile Dataset**: از ۱۱۶۳۱ گره همراه با ۱۷۰۹۱۸ لبه تشکیل شده‌است و مربوط به مقالات مرتبط با کروکودیل می‌باشد.

Dataset statistics			
	Chameleon	Crocodile	Squirrel
Nodes	2,277	11,631	5,201
Edges	31,421	170,918	198,493
Density	0.012	0.003	0.015
Transitvity	0.314	0.026	0.348

شکل ۴-۱: آمارگان مربوط به مجموعه‌داده

۴-۳ پیش‌پرادزش

پیش‌پردازش مجموعه داده در هر تحقیق داده‌ای، بسیار حیاتی است. این مرحله می‌تواند تأثیر قابل‌توجهی بر کیفیت و دقت مدل‌های یادگیری ماشین، پیش‌بینی‌ها، و تحلیل‌هایمان داشته باشد. در ادامه به شرح مراحال خوانش داده و اقدامات صورت گرفته جهت پیش‌پردازش، تصفیه و آماده‌سازی دادگان می‌پردازیم.

لازم به ذکر است که باتوجه به اهمیت دادگان و کاربرد و تحلیل آن‌ها، امروزه ابزارهای بسیاری برای سهولت کار با دادگان ارائه شده‌اند. زبان برنامه‌نویسی پایتون [۱] نیز یک زبان برنامه‌نویسی سطح بالا است که به سبب راحتی نوشتارش و ابزارها و کتابخانه‌های کاربردی که ارائه می‌دهد، یکی از گزینه‌های اول برنامه‌نویسی برای تحلیل دادگان است. ما نیز برای انجام این پروژه از زبان پایتون و کتاب‌خانه‌های کاربردی‌اش بهره بردیم.

۴-۳-۱ خوانش دادگان

همانطور که در ۴-۲-۶ ذکر شد، داده‌های مورد بررسی در قالب فایل‌هایی به قالب‌های CSV و JSON دریافت شدند که لازم است به قالبی مناسب خوانش تبدیل شوند تا بتوان عملیات‌های پردازشی و ریاضیاتی مختلف را بر آن‌ها اجرا کرد. ما نیز برای خوانش داده‌های مربوط به لبه‌ها و مقادیرهدف گره‌ها از دیتافریم [۲] کتابخانه‌ی پانداس [۳] استفاده کردیم که امکانات بسیاری را در عین سهولت برای ما فراهم می‌کند. هم‌چنین برای خوانش دادگان مربوط به ویژگی‌های گره‌ها در قالب [۴] از کتاب‌خانه json استفاده کردیم اما در مراحل بعدی مجددا پردازش‌هایی انجام دادیم و به قالب‌های مناسب‌تری درآوردیم.

۴-۳-۲ استخراج داده‌های پرت و کنارگذاشتن آن‌ها

داده‌های پرت (**Outliers**) به داده‌هایی اشاره دارند که از الگوی عمومی داده‌ها بیرون می‌افتند و به نوعی از دیگر داده‌ها متمایز می‌شوند. این داده‌ها عمدتاً به دلیل ویژگی‌های خاص و نادر یا به دلیل خطاها

[1] Python
[2] Dataframe
[3] Pandas
[4] Dictionary

در جمع‌آوری داده‌ها ایجاد می‌شوند. این موارد می‌توانند در تحلیل داده و مدل‌سازی مشکل‌هایی ایجاد کنند، به عنوان مثال:

- تحلیل نادرست: وجود داده‌های پرت می‌تواند به نتایج نادرست و بی‌اعتبار در تحلیل‌های داده منجر شود. مدل‌های آموزش دیده بر داده‌های پرت، در پیش‌بینی‌ها و تصمیم‌گیری‌ها در برابر با دادگان آزمون ممکن است کاملا صعیف عمل کنند..
- افزایش واریانس: وجود داده‌های پرت می‌تواند واریانس داده‌ها را افزایش دهد. این می‌تواند باعث شود که مدل‌ها به داده‌هایی که الگوهای عمومی را دنبال نمی‌کنند، حساس‌تر شوند.
- تخریب کارایی مدل‌ها: وجود داده‌های پرت در داده‌های آموزش می‌تواند مدل‌ها را به یادگیری الگوهای اشتباه سوق دهد. این مسئله می‌تواند به کاهش دقت پیش‌بینی‌ها و بهبودپذیری مدل‌ها منجر شود.

برای مدیریت داده‌های پرت به شکل زیر عمل می‌شود:

۱. تشخیص داده‌های پرت: با استفاده از روش‌های آماری و محاسباتی، داده‌های پرت شناسایی می‌شوند. درواقع گره‌هایی که مقادیر هدف خارج توزیع آماری غالب دارند شناسایی می‌شوند و شاخصشان در لیستی نگهداری می‌شود.

۲. حذف یا تصحیح داده‌های پرت: داده‌های پرت می‌توانند از مجموعه داده حذف شوند یا مورد تصحیح قرار گیرند.در این مرحله ما دادهای پرت را از مجموعه دادگان کنار گذاشتیم. به این صورت که مقادیر هدف، ارتباطات(لبه‌ها) و ویژگی‌های متناظر این گره‌ها را شناسایی و از دیتافریم‌ها و دیکشنری حذف کردیم. پس از این کار به طبع نیاز، شاخص‌ها را با مقادیر جدید(بدون وجود جای خالی شاخص‌های حذف شده) بروزرسانی کردیم تا در مراحل بعدی به مشکل برنخوریم.

۴-۳-۳ نرمال‌سازی مقادیر هدف

مقادیر هدف دادگان مورد بررسی ما، در بازه‌ی بسیار گسترده‌ای از اعداد طبیعی قرار می‌گیرند. برای مثال از ۱۵ تا ۵۰۸۹۵۷ برای مجموعه دادگان آفتاب‌پرست (با وجود دادگان پرت) که مقادیر بزرگ این داده می‌تواند مشکلات محاسباتی ایجاد کند ویا در یادگیری شبکه‌عصبی اختلال ایجاد کند. بنابراین ما با استفاده از روش کمینه-بیشینه[۵] عملیات نرمال‌سازی[۶] مقادیر هدف گره‌ها را انجام می‌دهیم و اعداد را به بازه‌ی ۰ تا ۱ می‌بریم. فرمول این روش به شکل زیر است:

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (۱-۴)$$

[5] Min-Max

[6] Normalization

در اینجا:

- $X_{normalized}$ نمایش دهنده مقدار نرمال‌سازی شده است.
- X نمایش دهنده مقدار اصلی داده است.
- X_{min} نمایش دهنده حداقل مقدار داده‌ها در مجموعه داده است.
- X_{max} نمایش دهنده حداکثر مقدار داده‌ها در مجموعه داده است.

۴-۳-۴ روش کدبندی وان-هات(One-Hot-Encoding)

تبدیل ویژگی‌های گره‌ها به بردارهای با مقادیر ۱ و ۰، یک روش متداول در تحلیل گراف و یادگیری ماشین است که به تحلیل داده‌های گرافی و استفاده از آنها در مدل‌های یادگیری عمیق کمک می‌کند. این کار برای چند دلیل مهم انجام می‌شود:

۱. تبدیل به داده عددی: بسیاری از الگوریتم‌های مدل‌های یادگیری ماشین نیاز به ورودی‌های عددی دارند. تبدیل کردن ویژگی‌ها به شکل بردارهای دودویی (با مقادیر ۱ و ۰)، یک راه برای تبدیل داده‌های گرافی غیرعددی به فرمتی مناسب برای مدل‌های یادگیری ماشین است.

۲. از دست رفتن ترتیب: بسیاری از ویژگی‌های اجزای گراف (مانند گره‌ها و یال‌ها) ترتیب خاصی ندارند ویا عدد متانظر با آن‌ها ارزش کمی ندارد بلکه وجود یا عدم وجود آن ویژگی برای ما دارای معنا است. تبدیل به بردارهای با مقادیر ۱ و ۰ این مشکل را حل می‌کند. به عبارت دیگر، ویژگی‌ها ترتیب خاصی ندارند و اگر برای آن‌ها اعداد متفاوت در نظر بگیریم، می‌تواند به معنای ترتیب و یا تفاوت در ارزش تعبیر شود و تحلیل و نتایج غیرواقعی حاصل کند. بنابراین این تکنیک یک اقدام واجب برای دادگان است.

۳. سهولت پردازش و تحلیل مدل: این تبدیل به مدل‌ها کمک می‌کند تا به آسانی ویژگی‌ها را تحلیل کنند. به عنوان مثال، مدل‌های عمیق که از شبکه‌های عصبی استفاده می‌کنند، می‌توانند به سادگی ویژگی‌های دودویی را درک کنند و از آنها برای انجام تصمیم‌گیری‌های پیچیده استفاده کنند.

به طور کلی، تبدیل ویژگی‌های گره‌ها به بردارهای با مقادیر ۱ و ۰، یک ابزار قدرتمند برای استفاده از داده‌های گرافی در مدل‌های یادگیری ماشینی و تحلیل داده‌هاست. و شکل معنادارتری به دادگان می‌دهد و فعالیت‌های یادگیری و تحلیل را ممکن می‌سازد. در مجموعه دادگان مورد بررسی ما نیز به هر کلمه یک عدد طبیعی نسبت داده‌شد و ویژگی‌های هر گره به صورت لیستی از مقادیر عددی(لیستی از اعداد طبیعی که نشان‌دهنده کلمات بکارگرفته شده در آن گره هستند) ارائه شدند. ما برای ساختن بردار ویژگی‌ها، برای هر گره یک بردار(به طول تعداد کل ویژگی‌های یکتا) ساختیم که در آن خانه‌ی متناظر

با هر ویژگی مقداری برابر با ۰ یا ۱ دارد. مقدار۱ برای یک ویژگی به معنای وجود و مقدار ۰ به معنای عدم وجودش در آن گره است. این بردارها در قالب Tensor ایجاد شدند و در مرحله بعد مورد استفاده قرار گرفتند.

۴-۴ آماده‌سازی نهایی

با پیش‌پردازش و آماده‌سازی اولیه داده‌ها، دادگان ما از حالت خام کمی خارج شدند و آمادگی بیش‌تری برای انجام محاسبات و پردازش‌ها پیدا کردند. اما هنوز به طور کامل، قابل درک برای مدل‌های یادگیری ماشین و شبکه‌های عصبی نیستند چراکه قالب استاندار و یکپارچه گرافی ندارند. هم‌چنین نیاز است تا یکسری لیست‌ها جهت دسته‌بندی دادگانمان به داده‌های آموزشی[۷]، اعتبارسنجی[۸] و آزمون[۹] ایجاد و به گراف اضافه کنیم.

۴-۴-۱ ایجاد گراف

برای اینکه از دادگان پردازش شده‌ی موجود بتوانیم گراف مورد نیاز را ایجاد کنیم از کتاب‌خانه‌ی dgl [۱۰] استفاده می‌کنیم که ابزار graph را در اختیار ما قرار می‌دهد. با تفکیک گره‌های مبدا و گره‌های مقصد (در دیتافریم لبه‌ها) و دادن آن به ابزار graph، ابتدا گراف را ایجاد می‌کنیم. سپس مقادیر هدف را به قالب Tensor درآورده و با عنوان "target" و بردارهای ویژگی‌ها را با عنوان "feature" به گراف اضافه می‌کنیم.

۴-۴-۲ ایجاد ماسک‌ها

در روند یادگیری و ارزیابی شبکه‌عصبی نیاز است که درواقع سه مجموعه داده داشته‌باشیم به نام‌های مجموعه دادگان آموزشی، اعتبار سنجی و تست. در واقع نیاز است دادگان اولیه خود را به این ۳ دسته تقسیم کنیم و در هر کدام بخشی از دادگان را قرار بدهیم که با یکدیگر همپوشانی نداشته باشند و کاملا مجزا از یکدیگر باشند. البته، نکته مهمی که باید در نظر داشته این است که این ماسک‌ها معمولاً به صورت بردارهای دودویی(همانند آنچه برای ویژگی‌ها داشتیم) مورد استفاده قرار می‌گیرند، به این معنی که هر داده به شکل ۱ یا ۰(یا True و False) علامت گذاری می‌شود تا نشان دهد کدام داده در هر مرحله مورد استفاده است. پس برای هر ماسک یک لیست(یا بردار) از True و Falseها(متناظر با هر گره) خواهیم داشت که وجود یا عدم وجود گره‌ها را برای آن ماسک نشان می‌دهد.

- **Train Mask** (ماسک آموزشی): این ماسک برای آموزش مدل‌های یادگیری ماشینی استفاده

[7] Train
[8] Validation
[9] Test
[10] Deep Graph Learning

می‌شود. همه داده‌هایی که به عنوان "۱" در این ماسک نشان داده می‌شوند، به عنوان داده‌های آموزش برای مدل‌ها استفاده می‌شوند و در فرایند یادگیری مدل شرکت می‌کنند.

- **Validation Mask** (ماسک اعتبارسنجی): این ماسک برای اعتبارسنجی عملکرد مدل‌ها استفاده می‌شود. داده‌هایی که به عنوان "۱" در این ماسک نشان داده می‌شوند، برای اعتبارسنجی و تنظیم پارامترهای مدل‌ها مورد استفاده قرار می‌گیرند. این مرحله به ارزیابی عملکرد مدل در داده‌هایی که قبلاً دیده نشده‌اند، کمک می‌کند و باتوجه به عملکرد مدل در برابر این داده‌ها پارامترها تنظیم می‌شوند.
- **Test Mask** (ماسک آزمون): این ماسک برای آزمون و ارزیابی نهایی مدل‌ها استفاده می‌شود. داده‌هایی که به عنوان "۱" در این ماسک نشان داده می‌شوند، برای ارزیابی عملکرد نهایی مدل‌ها به کار می‌روند. در این مرحله، مدل‌ها برای پیش‌بینی داده‌های جدید و نهایی استفاده می‌شوند و عملکرد مدل در برخورد با داده‌های جدید ارزیابی می‌شود.

ما برای تولید این ماسک‌ها، با یک نسبت خاص، از دادگان به صورت تصادفی انتخاب کردیم و آن‌ها را در ماسک‌های مختلف قرار دادیم. نهایتا این سه دسته را با عنوان‌های "train_mask" و "validation_mask" و "test_mask" به گراف اضافه کردیم.

فصل پنجم

پیاده‌سازی و مدل‌ها

۵-۱ مقدمه

با وجود دادگان پیش‌پردازش و آماده‌سازی شده، حال می‌توان پردازش و تحلیل‌های نهایی را به عمل آورد. برای تحقق این مسئله، در قدم اول نیاز است تا مدل‌های مورد نیاز را طراحی و پیاده‌سازی کنیم. در این پروژه، ما از مدل‌های مبتنی بر شبکه عصبی و به طور دقیق‌تر شبکه عصبی گرافی بهره بردیم که توضیحات اولیه و پیش‌نیاز آن‌ها در ۴-۲ و ۵-۲ به طور کامل داده شد و درک مسائل پیش‌رفته‌تر این فصل را تسهیل کرد. در این فصل به معرفی مدل‌های مورد استفاده، شرح ساختار و پیاده‌سازی‌شان می‌پردازیم.

۵-۲ GAT

۵-۲-۱ معرفی

شبکه‌های توجه گرافی[1] (GAT) نوعی معماری شبکه عصبی هستند که برای کار بر روی داده‌های ساختار یافته گراف طراحی شده‌اند و در سال ۲۰۱۷ معرفی شدند[۱۶]. ایده اصلی پشت GATها استفاده از مکانیسم‌های توجه برای تعیین و اختصاص وزن به گره‌های همسایه در یک گراف است. این وزن‌ها نشان‌دهنده اهمیت یا میزان ارتباط هر گره همسایه با گره هدف است. با تجمیع ویژگی‌های گره‌های همسایه بر اساس این وزن‌ها، GATها می‌توانند ساختار گراف محلی را ثبت کنند و ویژگی‌های متمایز هر گره را بیاموزند. درواقع، این معماری از لایه‌های خود-توجه با ماسک برای حل کاستی‌های روش‌های قبلی مبتنی بر پیچش‌های گراف یا مشابه آن‌ها استفاده می‌کند. با استفاده از تکنیک توجه، گره‌ها می‌توانند بر روی ویژگی‌های همسایگان خود تمرکز کنند و اطلاعات مهم را از آن‌ها استخراج کنند. ایده اصلی تکنیک توجه این است که به جای اینکه برای تصمیم‌گیری در مورد داده‌ها از همه ویژگی‌ها و اطلاعات موجود استفاده شود، توجه به اجزای معین یا مهم‌تری از داده‌ها جلب شود. این قابلیت به مدل امکان می‌دهد تا بدون نیاز به عملیات ماتریسی هزینه‌بر (مانند معکوس کردن) ویا وابسته به شناخت اولیه از گراف، وزن‌های مختلف به گره‌ها در یک همسایگی نسبت دهد.

۵-۲-۲ ساختار

ساختار یک شبکه توجه گرافی معمولاً شامل اجزای زیر است:

لایه ورودی

شبکه یک گراف(شامل گره‌ها و لبه‌ها) همراه با ویژگی‌ها(ویژگی‌های گره‌ها در کاربرد رگرسیون گره) را به عنوان ورودی می‌پذیرد.

[1] Graph Attention Networks

مکانسیم توجه

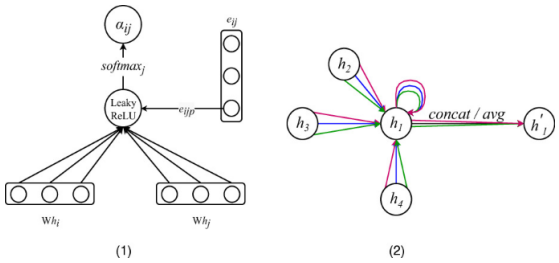
این قسمت بخش خاص و مهم GAT است. مکانیسم توجه وزن‌های توجه را برای هر گره بر اساس ویژگی‌های خود و همسایگانش محاسبه می‌کند. درواقع به‌جای اینکه مقادیر ساده‌ای را برای وزن‌دهی به همسایگان درنظر بگیرد، توابعی تعریف می‌کند که وزن گره‌ها با استفاده از آن و با توجه به ویژگی‌های گره‌ها محاسبه شود. این مکانیسم یاد می‌گیرد که کدام همسایگان برای هر گره مهم هستند. لازم به ذکر است که GAT می‌تواند از چندین سر توجه[۲] برای ثبت جنبه‌های مختلف روابط همسایگان استفاده کند. این قابلیت مدل را برای یادگیری الگوهای پیچیده در داده‌ها بهبود می‌دهد. برای محاسبه‌ی وزن‌های توجه می‌توان از شبکه عصبی(ضرایب و متغیرهای قابل یادگیری) همراه با اعمال تابع فعال ساز(مانند LeakyReLU) استفاده کرد. پس از محاسبه‌ی این توجه‌ها، تابع Softmax اعمال می‌شود تا این وزن‌ها(αها) نرمال‌سازی شوند و مجموعی برابر با ۱ داشته باشند.

لایه تجمیع

پس از محاسبه وزن‌های توجه، ویژگی‌های گره‌های همسایه برای هر گره تجمیع می‌شوند. این کار ارتباط‌ها و جریان اطلاعات در گراف را نشان می‌دهد.

لایه خروجی

لایه خروجی نهایی بسته به وظیفه ممکن است متفاوت باشد. برای رگرسیون گره‌ها، معمولاً شامل لایه‌های رگرسیون برای پیش‌بینی مقادیر هدف می‌شود.



شکل ۵-۱: ساختار و عملکرد GAT

۵-۲-۳ روابط و فرمول‌ها

معادلات کلیدی مورد استفاده در GAT در ادامه آمده است. مقدار توجه گره i به همسایه j:

$$e_{ij} = LeakyReLU\left(\vec{a}^T[W\vec{h}_i||W\vec{h}_j]\right) \quad (۱-۵)$$

[2] Attention Head

وزن‌های توجه نرمال‌سازی شده برای گره i و همسایگانش:

$$\alpha_{ij} = softmax(e_{ij}) = softmax(LeakyReLU(\vec{a}^T[W\vec{h}_i||W\vec{h}_j])) \quad (۲-۵)$$

درحالی که α_{ij} وزن توجه به گره j نسبت به گره i است. $\vec{a}$ یک بردار پارامتر قابل یادگیری است. $\vec{W}$ یک ماتریس وزن است. $\vec{h}_i$ و $\vec{h}_j$ ویژگی‌های جاسازی شده‌ی گره‌های i و j است و $||$ به معنای اتصال و ادغام دو بردار به هم است. خروجی نهایی گره i به صورت مجموع وزنی از ویژگی‌های همسایگان آن همراه با اعمال تابع غیرخطی ساز محاسبه می‌شود:

$$\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij} W\vec{h}_j\right) \quad (۳-۵)$$

در حالی که σ یک تابع فعال سازی است (به عنوان مثال، ReLU) و $\mathcal{N}(i)$ مجموعه گره های همسایه گره i است. در حالتی که چند سر توجه داشته باشیم، هرکدام را جداگانه محاسبه و همه را با هم ادغام می‌کنیم:

$$\vec{h}'_i = \|_{k=1}^{K}\sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k W^k\vec{h}_j\right) \quad (۴-۵)$$

۵-۲-۴ پیاده‌سازی

ما برای پیاده‌سازی این شبکه عصبی به شکل زیر عمل کردیم:

معماری مدل

مدل ما به شکل یک کلاس پایتون[۳] تعریف شده است که برای مقداردهی اولیه‌اش ابعاد لایه ورودی(تعداد ویژگی‌ها)، ابعاد لایه‌های پنهان، تعداد سرهای توجه و ابعاد لایه خروجی را دریافت می‌کند(ابعاد مورد استفاده قرارگرفته در بخش‌های بعدی همراه با نتایج آورده شده‌اند). سپس با استفاده از این اطلاعات مدل و لایه‌هایش را ایجاد می‌کند:

- لایه اول: لایه اول یک پیچش[۴] است با ابعاد تعداد ویژگی‌ها در تعداد ویژگی‌های پنهان، همراه با تعداد سرهای توجه دریافت شده. مقدار افت توجه[۵] برابر با 0.6 است.

[3] Python Class
[4] Convolution
[5] Attention Dropout

- لایه دوم: لایه دوم یک پیچش است با ابعاد تعداد ویژگی‌های پنهان * تعداد سرها در تعداد ویژگی‌های پنهان، همراه با تعداد سرهای توجه دریافت شده. مقدار افت توجه برابر با 0.6 است.
- لایه خروجی: باتوجه به کاربرد ما که رگرسیون است، نیاز است در لایه آخر یک تابع خطی ساز استفاده کنیم. برای این‌کار از یک لایه‌ی تماما متصل[۶] خطی‌ساز استفاده می‌کنیم.

عملیات روبه‌جلو

در این تابع[۷] درواقع پردازش برروی داده‌های ورودی با استفاده از لایه‌های تعریف شده در قسمت قبل صورت می‌گیرد و خروجی مدل به ازای مقادیر ورودی(ویژگی‌های گره‌های گراف) برای گره‌ها محاسبه می‌شود(عملیات پیش‌بینی[۸]). درواقع تابع گراف(شبکه‌ی گره‌ها و یال‌های بین‌شان) را همراه با بردارهای ویژگی‌های گره‌ها دریافت می‌کند و خروجی پردازش شده(مقادیر پیش‌بینی شده برای گره‌ها) را برمی‌گرداند. پردازش صورت گرفته به شرح زیر است:

- لایه اول: گراف و ویژگی‌هایش در اولین قدم به لایه اول داده‌می‌شوند و خروجی آن هموارسازی((flatten(1) می‌شود. این خروجی هموارسازی شده از یک تابع فعال ساز(ReLU) عبور می‌کند و حاصل به عنوان ورودی به لایه بعدی می‌رود. از تابع فعال‌ساز ReLU استفاده شد چراکه مسئله ما رگرسیون است و مقادیر محدود به بازه‌ای خاص و یا صفر و یک نیستند. پس توابع فعال‌سازی مانند Sigmoid نمی‌توانند مقادیر پیوسته خروجی را مدل کنند و بیش‌تر در مسائل دسته‌بندی و یا پیش‌بینی یال استفاده می‌شوند.
- لایه دوم: خروجی پردازش‌شده‌ی لایه اول را به عنوان ورودی دریافت می‌کند و حاصلش همانند لایه قبل هموارسازی می‌شود و از تابع فعال‌ساز عبور می‌کند.
- لایه خروجی: مقادیر حاصل از مرحله قبل را دریافت می‌کند و حاصل خطی شده را خروجی می‌دهد.

GATv2 ۳-۵

معرفی ۱-۳-۵

شبکه‌های توجه گرافی تکامل یافته‌اند و نسخه دوم، GATv2، نمایانگر تکامل این معماری است[۴]. GATv2 یک نوع شبکه عصبی است که برای پردازش داده‌های گرافی طراحی شده است و به عنوان

[6] Fully Connected
[7] Forward
[8] Prediction

نسخه بهبودیافته GAT اصلی معرفی شده است. این شبکه‌ها به دلیل کارآیی در وظایف گوناگون مبتنی بر گراف، محبوبیت چشمگیری به دست آورده‌اند.

۵-۳-۲ ساختار

ساختار GATv2 از همان مؤلفه‌های اصلی GAT پیروی می‌کند. شامل لایه‌های ورودی، میانی و خروجی همراه با سرهای توجه. اما شبکه‌های عصبی گرافی GAT انواع بسیار محدودی از توجه را محاسبه می‌کند و رتبه‌بندی امتیازات توجه در گره‌ها بدون قید و شرط خاصی است. این نوع توجه محدود به عنوان توجه ایستا تعریف می‌شود و از توجه پویا که اطلاعات بسیار بیش‌تری دارد، تمییز داده می‌شود. از آنجا که GAT از مکانیزم توجه ایستا استفاده می‌کنند، مشکلات گرافی ساده‌ای وجود دارد که این مدل‌ها نمی‌توانند بیان کنند. در یک مسئله کنترل شده، می‌توان نشان داد که توجه ایستا حتی از برازش داده‌های آموزشی توسط GAT جلوگیری می‌کند[۴]. برای حذف این محدودیت، یک راه حل ساده با تغییر ترتیب عملیات‌ها ارائه داده شده که GATv2 بیانگر آن است.

۵-۳-۳ روابط و فرمول‌ها

$$e_{ij} = \vec{a}^T LeakyReLU\left(W[\vec{h}_i||\vec{h}_j]\right) \tag{۵-۵}$$

۵-۳-۴ پیاده‌سازی

پیاده‌سازی این شبکه‌عصبی نیز به دلیل شباهت با GAT با همان ساختار صورت گرفته که در ۵-۲-۴ قابل دریافت است.

۵-۴ GCN

۵-۴-۱ معرفی

شبکه‌های پیچشی گرافی یا GCN مدل دیگری از شبکه‌های عصبی هستند که برای پردازش داده‌های با ساختار گراف طراحی شده‌اند[۱۰]. شبکه پیچشی گرافی، شبکه‌های عصبی پیچشی یا همان CNN را از شبکه‌های منظم کم بعد(که در آن داده‌هایی مانند تصویر، ویدیو و گفتار نمایش داده می‌شود) به حوزه‌هایی نامنظم از داده با ابعاد زیاد(مانند شبکه‌های اجتماعی ویا اتصالات مغزی) که ساختار گرافی دارند، تعمیم می‌دهد. این معماری فرمولی از CNNها را در زمینه تئوری گراف ارائه می‌دهد که زمینه

ریاضی لازم را برای طراحی فیلترهای پیچشی سریع بر روی گراف‌ها فراهم می‌کند. تکنیک پیشنهادی، برای هر ساختار گرافی قابل به‌کارگیری است. این امر درحالی برآورده می‌شود که همان پیچیدگی محاسباتی خطی و پیچیدگی یادگیری ثابت را مانند CNN های کلاسیک ارائه می‌دهد. در واقع، GCN با انتشار اطلاعات بین گره‌ها در یک گراف تعریف می‌شود. این روش بر اساس این ایده عمل می‌کند که بازنمایی گره را می‌توان با جمع‌آوری و تبدیل اطلاعات از همسایگانش به روز کرد. در زمینه رگرسیون گره، مدل‌های GCN یاد می‌گیرند که مقادیر گره را براساس این بازنمایی‌های به‌روزرسانی شده پیش‌بینی کنند.

۵-۴-۲ ساختار

این بخش به اجزای ساختاری و مکانیسم های یک لایهGCN می‌پردازد.

تجمیع

ابتدا اطلاعات ویژگی‌ها(جاسازی‌ها) از گره‌های همسایه جمع‌آوری می‌شود و با یک روش مانند میانگین‌گیری تجمیع و تبدیل به یک جاسازی می‌شود.

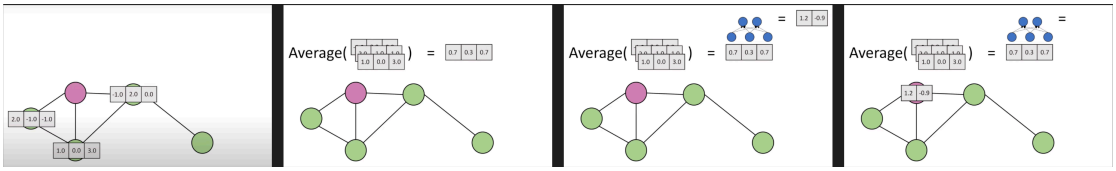
اعمال شبکه‌عصبی

حال جاسازی تجمیع شده از یک شبکه عصبی متراکم عبورد داده می‌شود. این به معنی اعمال ماتریس‌های وزن و توابع فعال‌ساز است. این کار برای هر گره جداگانه انجام می‌شود.

باتوجه به وظیفه، ساختار شبکه عصبی می‌تواند متفاوت باشد. برای مثال برای یک دسته‌بندی دودویی(مثلا اسپم بودن یا نبودن) می‌توان لایه آخر شبکه را از یک نورون ایجاد کرد.

برروزرسانی

نهایتا، جاسازی گره با خروجی شبکه عصبی بروزرسانی می‌شود.



شکل ۵-۲: ساختار و عملکرد GCN

۵-۴-۳ روابط و فرمول‌ها

این بخش به تشریح فرمول‌های ریاضی کلیدی درگیر در GCN می‌پردازد.

- تجمیع پیام: مرحله تجمیع پیام ترکیبی خطی از بردارهای ویژگی گره‌های همسایه را محاسبه می‌کند. در یک لایه استاندارد GCN، این مرحله به صورت زیر بیان می‌شود:

$$h_v^{(l+1)} = \sigma\left(\sum_{u\in\mathcal{N}(v)} \frac{1}{c_v} W^{(l)} h_u^{(l)}\right) \qquad (۶-۵)$$

درحالی که $h_v^{(l)}$ بردار ویژگی گره v در لایه l نشان می‌دهد. σ یک تابع فعال‌سازی را نشان می‌دهد (به عنوان مثال، ReLU). $\mathcal{N}(v)$ مجموعه گره‌های همسایه گره v را نشان می‌دهد. $W^{(l)}$ نشان دهنده ماتریس وزن برای l-امین لایه است. c_v یک ثابت نرمال‌سازی برای اطمینان از یک فرآیند آموزشی پایدار است.

- مقداردهی اولیه وزن‌ها: مقداردهی اولیه مناسب وزن‌ها، برای آموزش موثر GCNها بسیار مهم است. ماتریس‌های وزن را می‌توان با استفاده از روش‌هایی مانند مقداردهی اولیه Xavier یا He مقداردهی کرد.

۵-۴-۴ پیاده‌سازی

ما برای پیاده‌سازی این شبکه عصبی به شکل زیر عمل کردیم:

معماری مدل

مدل ما به شکل یک کلاس پایتون[۹] تعریف شده است که برای مقداردهی اولیه‌اش ابعاد لایه ورودی(تعداد ویژگی‌ها)، ابعاد لایه‌های پنهان و ابعاد لایه خروجی را دریافت می‌کند(ابعاد مورد استفاده قرارگرفته در بخش‌های بعدی همراه با نتایج آورده شده‌اند). سپس با استفاده از این اطلاعات مدل و لایه‌هایش را ایجاد می‌کند:

- لایه اول: لایه اول یک پیچش است با ابعاد تعداد ویژگی‌ها در تعداد ویژگی‌های پنهان.
- لایه دوم: لایه دوم یک پیچش است با ابعاد تعداد ویژگی‌های پنهان در ابعاد خروجی.
- لایه خروجی: باتوجه به کاربرد ما که رگرسیون است، نیاز است در لایه آخر یک تابع خطی ساز استفاده کنیم. برای این‌کار از یک لایه‌ی تماما متصل استفاده می‌شود که برای هر گره یک مقدار خروجی را تجمیع و برآورد می‌کند.

[9] Python Class

عملیات روبه‌جلو

در این تابع درواقع پردازش برروی داده‌ای ورودی با استفاده از لایه‌های تعریف شده صورت می‌گیرد و خروجی مدل به ازای مقادیر ورودی(ویژگی‌های گره‌های گراف) برای گره‌ها محاسبه می‌شود(عملیات پیش‌بینی). درواقع تابع گراف(شبکه‌ی گره‌ها و یال‌های بین‌شان) را همراه با بردارهای ویژگی‌های گره‌ها دریافت می‌کند و خروجی پردازش شده(مقادیر پیش‌بینی شده برای گره‌ها) را برمی‌گرداند. پردازش صورت گرفته به شرح زیر است:

- لایه اول: گراف و ویژگی‌هایش در اولین قدم به لایه اول داده‌می‌شوند و خروجی آن هموارسازی((flatten(1) می‌شود. این خروجی هموارسازی شده از یک تابع فعال ساز(ReLU) عبور می‌کند و حاصل به عنوان ورودی به لایه بعدی می‌رود.
- لایه دوم: خروجی پردازش‌شده‌ی لایه اول را به عنوان ورودی دریافت می‌کند و حاصلش همانند لایه قبل هموارسازی می‌شود و از تابع فعال‌ساز عبور می‌کند.
- لایه خروجی: مقادیر حاصل از مرحله قبل را دریافت می‌کند و حاصل خطی شده را خروجی می‌دهد.

GraphSAGE ۵-۵

۱-۵-۵ مقدمه

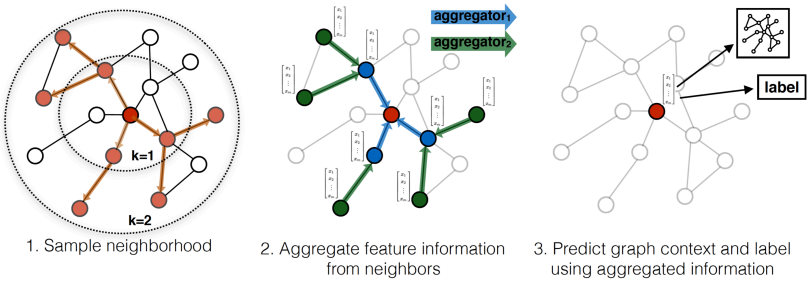
GraphSAGE، مخفف Graph Sample and Aggregated Embeddings، یک چارچوب قدرتمند برای ایجاد جاسازی گره‌ها در یک گراف است. برخلاف روش‌های سنتی که صرفاً بر ویژگی‌های گره یا ویژگی‌های ساختاری متکی هستند، GraphSAGE از ساختار همسایگی محلی یک گره و اطلاعات مربوط به ویژگی‌های آن‌ها برای تولید بازنمایی‌های غنی و آگاه از زمینه استفاده می‌کند.

ایده اصلی پشت GraphSAGE، نمونه‌برداری و جمع‌آوری اطلاعات از همسایگی یک گره برای یادگیری نمایشی برای آن گره است. این رویکرد به GraphSAGE اجازه می‌دهد تا اطلاعات ساختاری و ویژگی‌های یک گره را در گراف ثبت کند، و آن را به ابزاری ارزشمند برای کارهای مختلف یادگیری ماشین مبتنی بر گراف، از جمله دسته‌بندی گره، پیش‌بینی پیوند، و رگرسیون گراف تبدیل می‌کند.

۲-۵-۵ ساختار

نمونه‌برداری **K-Hop** از همسایگان

در GraphSAGE، فرآیند با نمونه‌برداری از یک همسایگی با اندازه ثابت در اطراف هر گره آغاز می‌شود. اندازه این همسایگی که معمولاً با k نشان داده می‌شود، تعیین می‌کند که چه مقدار از ساختار گراف در



شکل ۵-۳: ساختار و عملکرد Graph SAGE

هنگام ایجاد جاسازی‌ها در نظر گرفته می‌شود. مقادیر کوچکتر k اطلاعات محلی را دریافت می‌کنند، در حالی که مقادیر بزرگتر اطلاعات سراسری بیشتری را در بر می‌گیرند. GraphSAGE به چندین لایه سازماندهی شده است که هر لایه نمونه‌برداری و تجمیع گره را انجام می‌دهد. این لایه‌ها به‌طور متوالی روی هم قرار می‌گیرند و به مدل اجازه می‌دهند تا با افزایش عمق، اطلاعاتی را از محله‌های بزرگ‌تر بگیرد. خروجی یک لایه به عنوان ورودی لایه بعدی عمل می‌کند و مدل را قادر می‌سازد تا درک خود از زمینه گره را اصلاح کند و جاسازی‌هایی با سطوح بالاتر انتزاع ایجاد کند.[۸]

- عمق K: عمق همسایگان مورد بررسی را نشان می‌دهد. برای مثال $K = 1$ همسایگانی از گره را نشان می‌دهد که با فاصله یک یال از آن گره قرار دارند(گره‌های همسایه‌ی آن گره) و $K = 2$ گره‌های همسایه‌ای را نشان می‌دهد که با فاصله‌ی ۲ یال از آن گره قرار دارند(همسایه‌های همسایگان آن گره). مقدار K معمولا بیش از ۲ یا ۳ انتخاب نمی‌شود چراکه در آن صورت ویژگی‌های همه‌ی گره‌ها در ایجاد جاسازی‌های گره‌ها نقش پیدا می‌کند(اطلاعات از تمام گره‌ها جمع‌آوری می‌شد) و به این شکل جاسازی‌های گره‌ها شبیه به هم می‌شود. در حالی که هر گره باید جاسازی مختص خود و با توجه به شرایط محلی خود را داشته باشد.

- Hop: اگر در فرایند جمع‌آوری تمام گره‌های همسایه‌ی عمق Kام را درنظر بگیریم، تعداد همسایگان به صورت نمایی ممکن است افزایش پیدا کند و هم‌چنین ممکن است گره‌هایی تعداد همسایگان بسیار زیادی داشته باشند و جمع‌آوری اطلاعات این گره‌ها کاری بسیار هزینه‌بر خواهد بود. برای همین از یک Hop استفاده می‌شود که بیانگر تعداد نمونه‌هایی است که باید از همسایگان جمع‌آوری شود. برای مثال $Hop = 2$ به معنای این است که از همسایگان گره مورد نظر، اطلاعات ۲ گره همسایه(با انتخاب تصادفی) باید جمع‌آوری شود.

تجمیع

هنگامی که عملیات نمونه‌برداری از همسایگان انجام شد، GraphSAGE اطلاعات ویژگی را از گره‌های نمونه‌برداری شده جمع می‌کند تا یک جاسازی برای گره هدف ایجاد کند. این تجمیع[۱۰] با استفاده از

[10] Aggregation

توابع مختلف تجمع انجام می‌شود که شامل روش‌های میانگین(Mean)، LSTM-based و Pooling-based می‌شود.

- **Mean Aggregator**: میانگین بردارهای ویژگی گره‌های نمونه‌برداری شده در همسایگی را محاسبه می‌کند. این یک راه ساده و کارآمد برای ایجاد تعبیه گره‌ها ارائه می‌دهد.
- **LSTM Aggregator**: جمع‌کننده مبتنی بر LSTM از شبکه‌های حافظه کوتاه‌مدت بلند مدت برای گرفتن اطلاعات متوالی در همسایگی گره استفاده می‌کند.
- **Pooling Aggregator**: از تکنیک‌هایی مانند max-pooling یا sum-pooling برای گرفتن جنبه‌های مختلف اطلاعات از گره‌های نمونه استفاده می‌کند. Max-Pooling بر مرتبط‌ترین ویژگی‌ها تأکید می‌کند، در حالی که sum-pooling اطلاعات کلی محله را ضبط می‌کند.

ادغام

پس از جمع‌آوری اطلاعات از همسایگان، ادغام [11] صورت می‌گیرد. به این معنا که این اطلاعات جمع‌آوری شده با تعبیه‌ی قبلی گره ادغام می‌شود و جاسازی گره برروزرسانی می‌شود.

۵-۵-۳ روابط و فرمول‌ها

- **Mean Aggregator**

$$h_v^{(l+1)} = Mean\left(\{h_u^{(l)} \forall u \in \mathcal{N}(v)\}\right) \quad (۵-۷)$$

اینجا: - $h_v^{(l)}$ نشان‌دهنده جاسازی گره v در لایه l است. - $\mathcal{N}(v)$ مجموعه گره‌های همسایه گره v است. - $Mean(\cdot)$ میانگین بردارهای ویژگی ورودی را محاسبه می‌کند.

- **LSTM Aggregator**

تجمیع کننده مبتنی بر LSTM از یک شبکه LSTM برای جمع‌آوری اطلاعات استفاده می‌کند و برای گرفتن اطلاعات متوالی از جاسازی‌های گره‌های همسایه استفاده می‌کند:

$$h_v^{(l+1)} = LSTMAggregate\left(\{h_u^{(l)} \forall u \in \mathcal{N}(v)\}\right) \quad (۵-۸)$$

- **Pooling** جمع‌آوری حداکثری:

[11] Aggregation

$$h_v^{(l+1)} = MaxPooling\left(\{h_u^{(l)} \forall u \in \mathcal{N}(v)\}\right) \quad (۹-۵)$$

و برای جمع‌آوری مجموعات:

$$h_v^{(l+1)} = SumPooling\left(\{h_u^{(l)} \forall u \in \mathcal{N}(v)\}\right) \quad (۱۰-۵)$$

این فرمول‌ها استراتژی‌های مختلف تجمع مورد استفاده در GraphSAGE را نشان می‌دهند.

۵-۵-۴ پیاده‌سازی

ما برای پیاده‌سازی این شبکه عصبی به شکل زیر عمل کردیم:

معماری مدل

مدل ما به شکل یک کلاس پایتون[۱۲] تعریف شده است که برای مقداردهی اولیه‌اش ابعاد لایه ورودی(تعداد ویژگی‌ها)، ابعاد لایه‌های پنهان و ابعاد لایه خروجی را دریافت می‌کند(ابعاد مورد استفاده قرارگرفته در بخش‌های بعدی همراه با نتایج آورده شده‌اند). سپس با استفاده از این اطلاعات مدل و لایه‌هایش را ایجاد می‌کند:

- لایه اول: لایه اول یک پیچش است با ابعاد تعداد ویژگی‌ها در تعداد ویژگی‌های پنهان.
- لایه دوم: لایه دوم یک پیچش است با ابعاد تعداد ویژگی‌های پنهان در تعداد ویژگی‌های پنهان.
- لایه خروجی: لایه سوم یک پیچش است با ابعاد تعداد ویژگی‌های پنهان در یک، که به ازای هر گره، یک خروجی حاصل می‌کند.

عملیات روبه‌جلو

در این تابع درواقع پردازش برروی داده‌ای ورودی با استفاده از لایه‌های تعریف شده صورت می‌گیرد و خروجی مدل به ازای مقادیر ورودی(ویژگی‌های گره‌های گراف) برای گره‌ها محاسبه می‌شود(عملیات پیش‌بینی). درواقع تابع گراف(شبکه‌ی گره‌ها و یال‌های بین‌شان) را همراه با بردارهای ویژگی‌های گره‌ها

[12] Python Class

دریافت می‌کند و خروجی پردازش شده(مقادیر پیش‌بینی شده برای گره‌ها) را برمی‌گرداند. پردازش صورت گرفته به شرح زیر است:

- لایه اول: گراف و ویژگی‌هایش در اولین قدم به لایه اول داده‌می‌شوند و خروجی آن از یک تابع فعال ساز(ReLU) عبور می‌کند و حاصل به عنوان ورودی به لایه بعدی می‌رود.
- لایه دوم: خروجی پردازش‌شده‌ی لایه اول را به عنوان ورودی دریافت می‌کند و حاصلش همانند لایه قبل، از تابع فعال‌ساز عبور می‌کند.
- لایه خروجی: مقادیر حاصل از مرحله قبل را دریافت می‌کند و حاصل خطی شده را خروجی می‌دهد.

فصل ششم

ارزیابی و نتیجه‌گیری

۶-۱ مقدمه

پیاده‌سازی مدل‌ها بدون ارزیابی نتایج‌شان معنایی ندارد. پس از پیاده‌سازی و به‌کارگیری مدل‌ها نیاز است تا عملکردشان بررسی و ارزیابی شود و این کار هم به طور کلی مرتبط است با میزان نزدیکی (ویا فاصله) نتایج حاصل شده توسط مدل با نتایج هدف و مورد انتظار. در موضوع شبکه‌های عصبی و شبکه‌های عصبی گرافی روش‌های ارزیابی متفاوتی وجود دارد که هرکدام با توجه به کاربرد، نوع داده و یا نیاز، مورد استفاده قرار می‌گیرد. در این فصل ما چند روش مرتبط با رگرسیون را ذکر کرده و مدل‌هایمان را با کمک آن‌ها آموزش می‌دهیم و عملکردشان را بررسی و مقایسه می‌کنیم.

۶-۲ معیارهای ارزیابی

در ارزیابی عملکرد مدل‌های رگرسیونی، چند معیار متداول وجود دارد که ما از آنها برای اندازه‌گیری دقت پیش‌بینی‌های مدل‌ها استفاده می‌کنیم. در ادامه، چند معیار ارزیابی مهم را معرفی و توضیح می‌دهیم.

۶-۲-۱ Mean Squared Error (MSE)

یک معیار متداول برای ارزیابی عملکرد مدل‌های رگرسیون است. این معیار، میانگین مربعات تفاضل‌های بین مقادیر پیش‌بینی شده و مقادیر واقعی(مقادیر اصلی) متغیر هدف را اندازه‌گیری می‌کند. فرمول محاسبه معیار MSE به صورت زیر است:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

در اینجا n تعداد نقاط داده است. y_i مقدار واقعی (اصلی) نقطه داده i است. $\hat{y}_i$ مقدار پیش‌بینی شده توسط مدل برای نقطه داده i است.

۶-۲-۲ Root Mean Squared Error (RMSE)

معیاری شبیه به MSE است با این تفاوت که جذر میانگین مربعات خطاها را محاسبه می‌کند. این معیار، خطا را به واحد اولیه متغیر وابسته برمی‌گرداند و برای مقایسه‌ی نتایج با مقادیر واقعی به کار می‌رود. RMSE به صورت زیر تعریف می‌شود:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

۶-۲-۳ Mean Absolute Error (MAE)

یک معیار دیگر از خطا در پیش‌بینی‌های رگرسیونی است که از میانگین مقادیر مطلق خطاها استفاده می‌کند. MAE به صورت زیر تعریف می‌شود:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

۶-۲-۴ Mean Absolute Percentage Error (MAPE)

معیاری است که به صورت درصدی خطای مدل را نسبت به واقعیت می‌سنجد. این معیار به ویژه مفید است زمانی که ما قصد اندازه‌گیری دقت نسبی مدل‌ها نسبت به واقعیت را داریم اما به دلیل تقسیماتی که در رابطه‌اش وجود دارد، ممکن است موجب بروز اختلال شود. MAPE به صورت زیر تعریف می‌شود:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\%$$

۶-۳ عملکرد مدل‌ها

با درنظر گرفتن تمامی تعاریف و مفاهیم ارائه شده، اکنون می‌توان درک درستی از نتایج پردازشات داشت و عملکرد مدل‌ها، معیارهای ارزیابی و دیگر خصیصه‌های پروژه را ارزیابی و مقایسه کرد. در این بخش، به شرح نتایج مدل‌هایمان و مقایسه‌شان در حالت‌های مختلف می‌پردازیم.

۶-۳-۱ جزئیات آموزش

در این بخش به توضیح نحوه‌ی به کارگیری شبکه های عصبی گرافی در پروژه می‌پردازیم.

- تابع زیان: برای ارزیابی عملکرد مدل و به روزرسانی متغیرها(یادگیری) نیاز است که معیار و تابعی داشته باشیم تا میزان خطا در پیش‌بینی(فاصله‌ی مقدار پیش‌بینی شده از مقدار واقعی) را محاسبه کند. برای این کار چند تابع زیان معرفی و اعمال شده‌اند که نتایج عملکرد مدل‌ها در بخش‌های بعدی آمده است.
- بهینه‌ساز: برای برروزرسانی متغیرهای شبکه عصبی نیاز است تا از الگوریتم مناسبی(مانند کاهش گرادیان) استفاده کنیم. در این پروژه، بهینه‌ساز آدام[1] به کار گرفته شده و نرخ یادگیری برابر با 0.005 تنظیم شده‌است.

[1] Adam optimizer

- دوره‌ها: یادگیری شبکه عصبی در یک دوره آموزش اتفاق نمی‌افتد و نیاز است که مدل بارها مقدار متغیرهایش را بروزرسانی کند. اما از طرفی تعداد دوره‌های زیاد هزینه‌بر است و ممکن است باعث بیش‌برازش شود و مدل در برابر داده‌های آزمون به خوبی عمل نکند. در این پروژه تعداد دوره‌ها ۵۰۰ تنظیم شده، اما آموزش در دوره‌های پایین‌تر اتفاق می‌افتد و از جایی به بعد بیش برازش قابل مشاهده است.
- آموزش: پس از تکمیل مراحل قبل می‌توان روند آموزش را برای مدل انجام داد. برای این کار مدل در حالت آموزش می‌رود، گراف و ویژگی‌ها(دادگان آموزشی) را به عنوان ورودی می‌گیرد و برای هر گره مقدار پیش‌بینی می‌کند. سپس با استفاده از تابع زیان خطای مقدار پیش‌بینی شده با مقدار واقعی گره‌ها محاسبه می‌شود و عملیات پس‌انتشار انجام می‌شود تا مدل بروزرسانی شود. در بروزرسانی و بهینه‌سازی مدل، آدام نقش ایفا می‌کند.
- ارزیابی: پس از انجام عملیات آموزش، عملیات ارزیابی نیز انجام می‌شود تا از نحوه‌ی عملکرد مدل مطلع شویم و میزان کاهش خطاها را در هر دوره مشاهده گنیم.

۶-۳-۲ ارزیابی و مقایسه

با استفاده از معیارهای ارزیابی مختلف، عملیات یادگیری برای مدل‌ها انجام شد و عملکردشان نیز ارزیابی شد. در ادامه نتایج ارزیابی‌ها در سه مجموعه داده مورد بررسی ما آمده است:

chameleon dataset

نتایج برای مجموعه دادگان chameleon به صورت زیر است:

مدل/تابع زیان	MSE	RMSE	MAE	MAPE
GAT	0.024	0.1586	0.0897	10.101
GATv2	0.025	0.1652	0.1023	3.884
GCN	0.029	0.1716	0.1099	1.045
GraphSAGE	0.036	0.1919	0.1204	1.049

شکل ۶-۱: نتایج دادگان chameleon

با توجه به مقادیری که مدل در استفاده از هر تابع زیان پیش‌بینی کرد و هم‌چنین صعودی یا نزولی بودن روند آموزش، تابع زیان MSE عملکرد بهتر و نتایج قابل‌اتکاتری را ارائه کرد و برای ارزیابی قرار

گرفت. همانطور که مشخص است، تابع زیان MAPE به دلیل رابطه ریاضی که به کار می‌گیرد، در برخورد با دادگان ما بد عمل می‌کند و باعث اختلال در عملکرد می‌شود.

crocodile dataset و **squirrel dataset**

نتایج برای مجموعه دادگان crocodile به صورت زیر است:

مدل/تابع زیان	MSE
GAT	0.0014
GATv2	0.0016
GCN	0.00175
GraphSAGE	0.00179

شکل ۶-۲: نتایج دادگان crocodile

نتایج برای مجموعه دادگان squirrel به صورت زیر است:

مدل/تابع زیان	MSE
GAT	0.0045
GATv2	0.0046
GCN	0.0069
GraphSAGE	0.0070

شکل ۶-۳: نتایج دادگان squirrel

۶-۳-۳ نتیجه‌گیری

به طور کل، نتایج عملکرد مدل‌ها بسیار وابسته است به مواردی از جمله ساختار دادگان، ارتباطات گره‌ها، جنس و عمق ویژگی‌هایشان. می‌توان گفت که هر مدل در کاربردهای مختلف، نقاط ضعف و قدرت ویژه خود را دارد و در مجموعه‌دادگانی خاص می‌تواند عملکردی جدید از خود نشان دهد. برای مثال شبکه پیچشی گرافی در یادگیری از دادگان کم‌حجم می‌تواند عملکرد خوبی از خود نشان دهد. از طرفی

شبکه‌های توجه گرافی با استفاده از مکانیزم توجه می‌توانند ویژگی‌های مخفی مفیدی از گره‌های اطراف استخراج کنند. با توجه به آزمایشات و نتایج علمی که از مدل‌ها جمع‌آوری شد، مشاهده شد که اگرچه اعداد و ارقام مربوط به ارزیابی این مدل‌ها نزدیک به هم بود، به طور کل شبکه‌های توجه گرافی (شبکه توجه گرافی معمولی و سپس نسخه دو) عملکرد بهتری از خود نشان دادند. سپس شبکه‌های پیچشی گرافی و نهایتا GraphSAGE عملکرد مناسبی از خود نشان دادند.

۶-۴ جمع‌بندی

این پایان‌نامه را با توضیح صورت مسئله و اهداف پروژه آغاز کردیم. فصل دو به ادبیات مسئله اختصاص داشت. در این فصل، مفاهیم پایه و دانش پیش‌نیاز برای حل مسئله را نام بردیم و شرح دادیم. در ادامه، برای آشنایی با تاریخچه‌ی تحلیل گراف‌ها، اشاره‌ای به روش‌های پیشین داشتیم. برخی از روش‌های سنتی رگرسیون در گراف‌ها معرفی کردیم و رویکردشان به اختصار توضیح داده‌شد. در فصل چهار، از اهمیت مجموعه دادگان در پردازش‌هایمان صحبت کردیم و ویژگی‌های مدنظر برای انتخاب یک مجموعه داده مناسب را ذکر کردیم. مجموعه دادگان انتخابی ما مقادیری خام داشت که به هیچ عنوان مناسب شبکه‌های عصبی نبود. پس، روش‌های مختلف پیش‌پردازش را پیاده‌سازی کردیم و با استفاده از این روش‌ها، دادگانی استاندارد و تصفیه‌شده تولید کردیم. با داشتن دادگان پیش‌پردازش شده، تنها باید مدل‌های شبکه عصبی گرافی را اعمال کنیم. پس در فصل پنج، همراه با توضیح ساختار مدل‌ها، آن‌ها را پیاده‌سازی کردیم و نهایتا در فصل شش نتایج‌شان را ارزیابی کردیم. دیدیم که در کاربرد ما و مجموعه‌دادگان ویژه مسئله، کدام شبکه‌های عصبی گرافی عملکرد مناسبی داشتند و مقادیر زیان هرکدام به چه صورت بود.

کتاب‌نامه

[1] ZINC 15 database. http://zinc15.docking.org. Accessed: [Insert Access Date].

[2] Belkin, M. and Niyogi, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, 2003.

[3] Borgelt, C. Efficient implementations of apriori and eclat. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning (ICML'03)*, 2002.

[4] Brody, Shaked, Alon, Uri, and Yahav, Eran. How attentive are graph attention networks?, Jan 2022.

[5] Duan, L., Tsang, I., Xu, D., and Chua, T. Domain adaptation from multiple sources via auxiliary classifiers. In *ICML*, 2009.

[6] Dwivedi, Vijay Prakash, Rampášek, Ladislav, Galkin, Mikhail, Parviz, Ali, Wolf, Guy, Luu, Anh Tuan, and Beaini, Dominique. Long range graph benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[7] Fouss, F., Pirotte, A., Renders, J. M., and Saerens, M. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(11):1427–1443, 2007.

[8] Hamilton, William L., Ying, Rex, and Leskovec, Jure. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 2017.

[9] Hu, Weihua, Fey, Matthias, Ren, Hongyu, Nakata, Maho, Dong, Yuxiao, and Leskovec, Jure. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021.

[10] Kipf, Thomas N. and Welling, Max. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017.

[11] Liu, Y., Li, Z., Pan, S., Gong, C., Zhou, C., and Karypis, G. Anomaly detection on attributed networks via contrastive self-supervised learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33:2378–2392, 2022.

[12] Lü, L. and Zhou, T. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[13] Luan, S., Hua, C., Lu, Q., Zhu, J., Chang, X., and Precup, D. When do we need gnn for node classification? 2022.

[14] Rozemberczki, Benedek, Allen, Carl, and Sarkar, Rik. Multi-scale attributed node embedding, 2019.

[15] Sandryhaila, A. and Moura, L. Discrete signal processing on graphs. *IEEE Transactions on Signal Processing*, 61(7):1644–1656, 2013.

[16] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lió, P., and Bengio, Y. Graph attention networks. 2017.

[17] Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

[18] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. A comprehensive survey on graph neural networks. *Ieee Transactions on Neural Networks and Learning Systems*, 32:4–24, 2021.

[19] Wu, Zhenqin, Ramsundar, Bharath, Feinberg, Evan N, Gomes, Joseph, Geniesse, Caleb, Pappu, Aneesh S, Leswing, Karl, and Pande, Vijay S. Moleculenet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

[20] Zhang, Z., Chen, D., Wang, J., Bai, L., and Hancock, E. R. Quantum-based subgraph convolutional neural networks. *Pattern Recognition*, 88:38–49, 2019.

[21] Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. Learning with local and global consistency. In *NIPS*, 2004.

[22] Zhu, X. and Ghahramani, Z. Learning from labeled and unlabeled data with label propagation. 2002.