پاییز ۱۴۰۰، دوره ۶، شماره ۳، پیاپی ۲۲
DOR :20.1001.1.2538161.1400.6.3.6.8

دریافت مقاله: ۱۳۹۹/۱۲/۱۸
پذیرش مقاله: ۱۴۰۰/۰۳/۲۵

نوع مقاله: پژوهشی

یک چارچوب برای پیش‌بینی پیوند با استفاده از نشانده و شبکه عصبی هم‌آمیختی

ابوالفضل شریفی*
دانشجوی کارشناسی ارشد، مهندسی نرم افزار، دانشکده مهندسی برق و کامپیوتر، دانشگاه کاشان، کاشان، ایران
پست الکترونیکی: sharifi.abolfazl@grad.kashanu.ac.ir

حمید مغانلو
دانشجوی کارشناسی ارشد، مهندسی نرم افزار، دانشکده مهندسی برق و کامپیوتر، دانشگاه کاشان، کاشان، ایران
پست الکترونیکی: h.moghanloo @grad.kashanu.ac.ir

فرشته زندی
دانشجوی کارشناسی ارشد، مهندسی نرم افزار، دانشکده مهندسی برق و کامپیوتر، دانشگاه کاشان، کاشان، ایران
پست الکترونیکی: f.zandi @grad.kashanu.ac.ir

مهدی وحیدی‌پور
استادیار، گروه مهندسی کامپیوتر، دانشکده مهندسی برق و کامپیوتر، دانشگاه کاشان، کاشان، ایران.
پست الکترونیکی: vahidipour@kashanu.ac.ir

چکیده

امروزه استفاده از نشانندهٔ[1] گره‌های شبکه، کاربردهای بسیاری را در تحلیل شبکه‌های پیچیده پیدا کرده است؛ نشانندهٔ یک گره، برداری است که این گره را در یک فضای جدید برداری نمایش می‌دهد. یافتن یک نمایش برداری مناسب برای گره‌های شبکه را یادگیری بازنمایی شبکه می‌نامند که در آن گره‌های شبیه در شبکه باید چنان در فضای برداری نمایش داده شوند که در آن فضا نیز به هم شبیه باشند و فاصله میان نشانندهٔ گره‌های شبیه در فضای جدید کم باشد. Node2vec یکی از روش‌های رایج برای یافتن نشانندهٔ گره‌های شبکه است. با فرض داشتن نشانده‌ها، مسئله تحلیل در شبکه‌های پیچیده می‌تواند تبدیل به مسئلهٔ دیگری در فضای برداری شود. در این مقاله، یک چارچوب سه مرحله‌ای، با نام DenseNet-LP پیشنهاد می‌شود که در آن مسئله پیش‌بینی پیوند در شبکه‌های پیچیده با مسئله رده‌بندی در فضای برداری جابجا می‌شود. در مرحله اول، نشانندهٔ گره‌ها با روش Node2vec به‌دست می‌آید. در مرحلهٔ دومِ DenseNet-LP، با استفاده از نشانده‌ها، به ازای هر گره یک ماتریس ساخته می‌شود تا در مرحله بعدی مورد استفاده قرار گیرد. در مرحله آخر DenseNet-LP، ماتریس مرتبط با دو گره متفاوت به یک شبکه عصبی داده می‌شود تا مسئله رده‌بندی را حل کند؛ آیا میان این دو گره پیوند وجود

* نویسندهٔ مسئول

1- embedding

دارد (رده اول) یـا وجـود نـدارد (رده دوم)؟ همچنین در این مقاله، در کنار مقایسـه روش‌هـای متفاوت یادگیری بازنمایی شبکه با Node2vec، نسخه جدیدی از این روش نیز پیشنهاد شده اسـت که کارآیی بالاتری در حل مسئله پیش‌بینی پیوند دارد.

واژه‌های کلیـدی : پیش‌بینی پیوند ، یادگیری بازنمایی، نشاننده، Node2vec استاندارد، Node2vec تعمیم‌یافته.

۱- مقدمه

مسئله پیش‌بینی پیوند یعنی تعیین وجود پیوند بین دو گره در شبکه و در آینده است. پیش‌بینی پیوند می‌تواند در شبکه‌های اجتماعی، شبکه‌های مشتریان، شبکه‌های مربوط به مقالات و همکاری نویسندگان و ... تعریف شود. مسئله پیش‌بینی پیوند را می‌توان با دو رویکرد مبتنی بر تشـابه و مبتنـی بر یادگیری حل کرد [۱]. روش مبتنی بر تشـابه فرض می‌کند که گره‌ها هرچه شـباهت بیشـتری داشـته باشند، امکان پیوند بین آن‌ها بیشتر است [۲]. این روش‌ها از اطلاعات محلی و شـباهت مسـیر برای پیش‌بینی پیوند استفاده می‌کنند. روش‌های مبتنی برتشابه از اطلاعات محلی تنها سـاختار محلی را در نظر می‌گیرند، مانند همسایگان مشترک بین دوگره. روش مبتنی بر مسیر از اطلاعات کلی شبکه استفاده می‌کند. مشـکل مبتنی بر شباهت این است که به توپولوژی شبکه متکی است و همچنین روش مبتنی بر سـاختار چندان قابل اعتماد نیست [۱]؛ زیرا ممکن است شـبکه‌های مختلف دارای خوشـه‌بندی و طول مسیرهای متمایز باشند امّا توزیع درجه آن‌ها یکسان است.

الگوریتم‌های مبتنی بر یادگیری، به دلیل دقت بالا بیشتر مورد استفاده قرار می‌گیرند [۳]. در این روش‌ها استخراج ویژگی‌ها از شـبکه پیچیده مورد توجه اسـت [٤]. یکی از روش‌های اسـتخراج ویژگـی، اسـتفاده از الگوریتم‌های یادگیری بازنمایی شبکه[۲] اسـت [٥]. در این روش‌ها برای هر گره در شبکه یک بردار ویژگی، به نام نشاننده ساخته می‌شود؛ نشـاندهٔ یک گره برداری است که این گره را در یک فضای جدیـد ویژگی نمایش می‌دهد. یافتن یک نمایش برداری مناسـب برای گره‌های شبکه باید به‌گونه‌ای باشد که در آن گره‌های شبیه به‌هم در شبکه، در فضای برداری ویژگی نیز به همدیگر نزدیک باشـند. یک روش بازنمایی شـبکه این الزام را به‌صورت یک مسـئله بهینه‌سازی در نظـر می‌گیرد که نتیجه حل آن یافتـن بازنمایی جدیدی از شبکه در یک فضای جدید است؛ بازنمایی از فضای گراف به فضای برداری ویژگی. DeepWalk[٦] و Node2vec[۷] و Graph2vec[۸] سـه الگوریتم شناخته شده در بازنمایی شبکه‌ها هستند.

بـا بازنمایی گره‌های یک شـبکه در فضـای برداری، مسایل مختلف تحلیل شبکه‌های پیچیده می‌تواند به مسایل جدیدی در فضای ویژگی تبدیل شـود. در فضای برداری ویژگـی، الگوریتم‌های مختلف و مدل‌هـای متفاوت برای رده‌بندی، خوشـه‌بندی، رگرسـیون و بهینه‌سازی وجود دارد. با تبدیل فضا می‌توان از تمامی آن‌ها اسـتفاده کرد. کافی اسـت یک مسئله تحلیل در شـبکه‌های پیچیده، مانند مسـئله پیش‌بینی پیوند را به یک مسئله متناظر در فضای برداری ویژگی، مانند رده‌بندی تبدیل نمود به‌طوری که حل مسـئله پیش‌بینی پیوند در فضای شبکه پیچیده متناظر با حل مسئله رده‌بندی میان نشاننده‌های گره‌ها باشد.

یکی از مدل‌های حل مسئله در فضای ویژگی شبکه‌های عصبی است. اسـتفاده از شبکه‌های عصبی در حوزه‌های مختلـف روزبهروز در حال افزایش اسـت. توسـعه‌های مختلف شبکه‌های عصبی کمک می‌کنند تا دسته‌های مختلفی از مسـائل مربوط به یادگیری ماشـین ماننـد رده‌بندی، خوشه‌بندی و رگرسیون حل شود.

شـبکه‌های عصبی عمیق[۳] بـه دلیل یادگیـری و بیان ویژگی‌های مناسـب در طبقه‌بندی‌هـای گوناگون از جمله تصاویر و شناسایی هدف، پیشرفت چشمگیری داشته‌اند. هر چه شبکه‌های عصبی مصنوعی عمیق‌تر شوند یادگیری بهتر می‌شود [۹]؛ امّا عمیق‌تر کردن شبکه مشکلاتی را ایجاد خواهد کرد. برای رفع چنین مشـکلاتی، شبکه‌های عصبی

2- Network Representation Learning

3- Deep Artificial Neural Networks

مصنوعی با مفاهیم جدیدی از ارتباطات میان لایه‌ها ایجاد شد. شبکه‌های عصبی هم‌آمیختی متراکم[4] یا به اختصار DenseNetها از جمله این شبکه‌های جدید هستند [۱۰].

در این مقاله، یک چارچوب سه مرحله‌ای با نام DenseNet-LP پیشنهاد شده است. این چارچوب برای حل مسئله پیش‌بینی پیوند در شبکه‌های پیچیده پیشنهاد می‌شود. سه مرحلهٔ DenseNet-LP به شرح زیر است:

- در مرحله اول با استفاده از روش یادگیری بازنمایی شبکه به ازای هر گره یک نشاننده (بردار ویژگی) ساخته می‌شود.
- در مرحله دوم بر اساس نشاننده‌های به‌دست آمده، به ازای هر گره یک نمایش ماتریسی به‌دست می‌آید. این ماتریس‌ها برای مرحله سوم مورد نیاز است.
- در مرحله سوم از یک شبکه عصبی هم‌آمیختی متراکم استفاده می‌شود. این شبکه عصبی یک مسئله رده‌بندی را حل می‌کند؛ ماتریس‌های ساخته شده (در مرحله دوم) مرتبط با یک جفت گره به شبکه عصبی داده می‌شود و این شبکه یا پیوندی میان آن دو گره را پیش‌بینی می‌کند (رده اول) و یا پیوند را پیش‌بینی نمی‌کند (رده دوم). بدیهی است این شبکه عصبی قبل از استفاده، با جفت گره‌های موجود در شبکه آموزش دیده است. اگر میان دو گره متفاوت پیوندی در شبکه پیچیده وجود داشته باشد (نباشد) یک نمونه آموزشی از رده اول (رده دوم) در نظر گرفته می‌شود.

در این مقاله، در مراحل اول و دوم روش‌های متفاوت و مختلفی استفاده شده است تا کارآیی چارچوب پیشنهادی در حل مسئله پیش‌بینی پیوند آزمایش شود. در همین راستا، یک روش جدید بازنمایی شبکه مبتنی بر روش Node2vec استاندارد هم پیشنهاد شده است (به نام Node2vec تعمیم‌یافته). آزمایش‌ها نشان می‌دهد Node2vec تعمیم‌یافته در حل مسئله پیش‌بینی پیوند کارآیی بالاتری نسبت به روش Node2vec استاندارد دارد.

بنابراین در این مقاله دو نوآوری اساسی ارایه شده است: ۱) ارائه یک الگوریتم Node2vec جدید به نام Node2vec تعمیم‌یافته و ۲) ارائه چارچوب سه‌مرحله‌ای DenseNet-LP برای حل مسئله پیش‌بینی پیوند .

ادامه مقاله به شرح گفته شده سازماندهی شده است. در بخش ۲ مفاهیم پایه‌ای مانند یادگیری بازنمایی، الگوریتم node2vec و شبکه‌های عصبی هم‌آمیختی متراکم به‌صورت اختصار مرور می‌شوند. در بخش ۳ الگوریتم پیشنهادی Node2vec گفته می‌شود. در بخش 4 DenseNet-LP بیان می‌شود. بخش ۵، بخش آزمایش‌ها است و در انتها بخش ششم به نتیجه‌گیری مقاله می‌پردازد.

۲- مفاهیم پایه

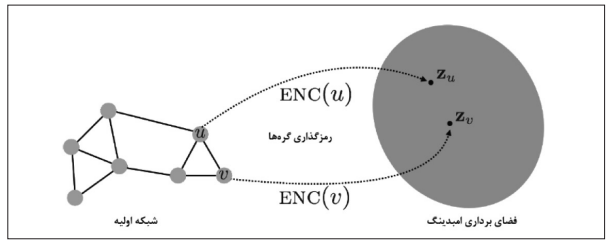
در ابتدای این بخش، یادگیری بازنمایی شبکه بررسی می‌شود. سپس، شرح مختصری از Node2vec استاندارد، به همراه انواع جستجوهای آن مرور می‌شوند. در انتهای این بخش، شبکه‌های عصبی هم‌آمیختی متراکم به اختصار بیان می‌شوند.

۲-۱- یادگیری بازنمایی شبکه

محققان علوم شبکه برای استخراج ویژگی‌ها از شبکه‌های پیچیده به اکتشاف ویژگی‌های کاربردی می‌پردازند. استخراج ویژگی‌های کاربردی منجر به یافتن بهتر روابط در میان شبکه‌ها می‌شود؛ که نتیجه نهایی آن حل کارآمد مسائل مختلف در شبکه‌های پیچیده مانند مسئله پیش‌بینی پیوند است. استخراج ویژگی‌ها وظیفهٔ مهندس ویژگی است. در سال‌های اخیر تمایل به یافتن ویژگی‌ها بدون نیاز به کمک مهندس ویژگی شدت یافته است. یادگیری بازنمایی شبکه استخراج و انتخاب ویژگی‌های کاربردی را به‌صورت خودکار انجام می‌دهد.

بازنمایی شبکه به منزله انتقال گره‌های شبکه به یک فضای جدید است. ایجاد بردار ویژگی (یا نشاننده) برای هر گره در شبکه را رمزگذاری می‌نامند. همان‌طور که در شکل ۱ نشان داده شده است به ازای هر گره u در شبکه اولیه، یک بردار z_u در فضای برداری نشاننده ایجاد می‌شود. وظیفه ساخت بردارهای نشاننده برای همه گره‌ها

4- Dense Convolutional Neural Networks



شکل ۱: انتقال گره‌ها به فضای برداری

بر عهده تابع رمزگذار ENC است. اجـرای این تابع برای تمامی گره‌های شبکه ۱ بازنمایی از شبکه اولیه در فضای برداری نشاننده ایجاد می‌کند. امّا سوال اصلی این است که کدام بازنمایی بهتر است؟

بازنمایی گره‌ها در فضای نشـاننده، توسط تابع ENC، باید چنان انجام شـود که بردارهای z_u و z_v که مربوط به دو گره همسایه در شبکه اولیه هستند، بسیار به هم نزدیک باشـند. فرض کنید ما از ضرب داخلـی دو بردار z_u و z_v برای اسـتنباط نزدیکی دو بردار اسـتفاده کنیم. بنابراین، شـباهت میان گره‌های u و v در شبکه اولیه باید متناظر با ضرب داخلی بردارهای z_u و z_v در فضای برداری نشاننده باشد. در یادگیری بازنمایی شبکه به دنبال تابع رمزگذاری هستیم که به بهترین شکل این هدف را در نظر بگیرد.

در یافتن نشـاننده‌ٔ گره، یکی از مسـایل اساسی یافتن گره‌های مشابه است. برای شناسـایی گره‌های شبیه یک گره u می‌توان از قدم‌زدنی استفاده کرد. از گره u قدم‌زنی شروع می‌شود و به یکی از گره‌های مجاور وارد می‌شود. در گام بعدی، قدم‌زنی از آن گره خارج و به یک گره مجاور بـه آن وارد می‌شـود. تعداد گام‌ها در طـول یک قدم‌زنی معمولاً پایین اسـت امّا تعداد باری که عمل قدم‌زنی از گره ابتدایی u انجام می‌شـود بالا اسـت. بدین‌ترتیب گره‌هایی که در طول قدم‌زنی‌های مکرر بازدید شـده‌اند مجموعه‌ای تشکیل می‌دهند که شـبیه گره u هستند. حال در یادگیری بازنمایی شبکه، نشاننده‌ٔ گره u باید نزدیک به نشاننده‌های گره‌های این مجموعه باشد.

راهبرد در قدم‌زنی، نحوه انتخاب مقصد بعدی قدم‌زنی اسـت. یکـی از سـاده‌ترین راهبردهای قدم‌زنـی، روش تصادفی است. در ادامه این روش تعریف می‌شود.

قدم‌زدن تصادفی: با توجه به یک گره به نام گره منبع u ، یک قدم‌زدن تصادفی با طول ثابت L شبیه‌سازی می‌شود. فرض کنید که C_i گره iام در مسیر حرکتی باشد که از گره $C_0 = u$ شروع می‌شود. گره‌های C_i با توزیع معادله (۱) انتخاب می‌شـوند. در این معادله، احتمال آن که گره بعدی در مسـیر قدم‌زنی گره x باشد، $c_i = x$ ، به شرطی تعیین می‌شود که گره جاری v است، $c_{i-1} = v$. گره x که یکی از همسایگان گره جاری v است، $(v, x) \in E$. در معادله (۱)، π_{vx} احتمال انتقال نرمال شـده بین گره‌های v و x و z نیز یک عدد نرمال شده ثابت است.

$$P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{z} & \text{if}(v, x) \in E \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

اگـر در راهبرد قدم‌زنی انتخاب مقصـد به طور کاملا تصادفی باشـد، آنگاه قدم‌زنی سـوگیری[5] شـده نیست. برخـلاف قدم‌زنی تصادفـی، راهبردهای پایـه‌ای دیگری وجود دارند که سوگیری‌شـده هسـتند. مانند راهبردهای BFS و DFS (شـکل ۲). از آنجـا که ایـن دو مفهوم پایه‌ای هستند، فرض بر آشنایی خواننده است. برای مطالعه بیشتر به [۱۱] مراجعه کنید.

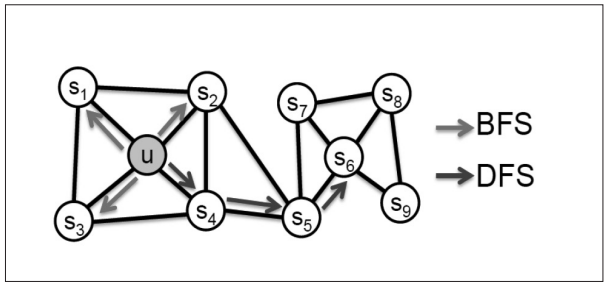
در تمامی راهبردهای ذکر شده، مقصد بعدی قدم‌زنی را گره جاری مشخص می‌کند. به عبارت رایج، دنباله گره‌هایی که قدم‌زنـی تولید می‌کند یک دنباله مارکـوف مرتبه اول است. امّا راهبرد می‌تواند قدم‌زنی تصادفی سوگیری‌شده مرتبـه دوم باشـد. بدان معنا که مقصد جدیـد قدم‌زنی را دو گـره اخیـر در دنباله تعیین می‌کنـد. در ادامه پیمایش سوگیری‌شده بر این اساس تعریف می‌شود.

۲-۱-۱- قدم‌زنی تصادفی سوگیری‌شده مرتبه دوم α

استفاده از پیمایش سوگیری‌شده در قدم‌زدن تصادفی می‌تواند مسـیر متفاوتی را تولید کند. ساده‌ترین پیمایش سوگیری‌شـده آن اسـت که از وزن پیوند میان گره‌های همسـایه x و v، یعنی w_{vx}، به جای π_{vx} اسـتفاده شـود، $\pi_{vx} = w_{vx}$.

اگـر تنها از مقادیـر وزن پیوندها برای این محاسـبه

5- bias



شکل ۲: انواع جستجوی DFS و BFS

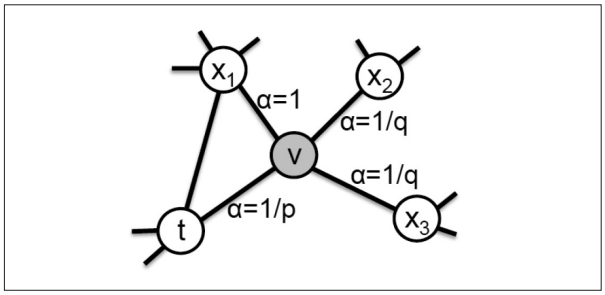
استفاده شود تنها ساختار شبکه است که در انجام قدم‌زدن تصادفی تأثیر دارد. برای رفع این مشکل از پیمایش سوگیری‌شده استفاده می‌شود. برای این مهم از رابطهٔ (۲) استفاده می‌شود که در آن از مقادیر سوگیری α برای محاسبه π_{vx} استفاده می‌شود.

$$\pi_{vx} = \alpha_{pq}(t,x).\, w_{vx} \qquad (۲)$$

برای توضیح رابطهٔ (۲) فرض کنید که در قدم‌زدن تصادفی از t به v حرکت شده است و حال باید برای مشخص کردن گام بعدی محاسبات از گره v انجام شود البته با درنظر گرفتن گره t (در گام بعدی، یک گام از v به x انجام می‌شود). این ارزیابی باعث محاسبه مقدار $\alpha_{pq}(t,x)$ می‌شود. برای محاسبه این مقدار سوگیری از دو پارامتر p و q استفاده می‌شود. پارامترهای p و q قدم‌زدن تصادفی را هدایت می‌کنند. به رابطهٔ (۳) دقت کنید که در آن $\alpha_{pq}(t,x)$ محاسبه می‌شود. d_{tx} کوتاه‌ترین فاصله بین گره t و گره x است؛ قدم‌زدن تصادفی از پیوند (t,v) عبور کرده و اکنون در گره v قرار دارد و برای انجام گام بعدی باید گره x انتخاب شود.

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases} \qquad (۳)$$

در رابطهٔ (۳)، پارامتر p احتمال بازدید مجدد و بلافاصله گره t را کنترل می‌کند. اگر p کمتر از $\min(q,1)$ باشد احتمال آن که در گام بعدی گره t مجدد بازدید شود بیشتر است. (در گام بعدی از گره v به گره t برمی‌گردد) اگر مقدار p بیشتر از $\max(q,1)$ باشد احتمال نمونه‌برداری



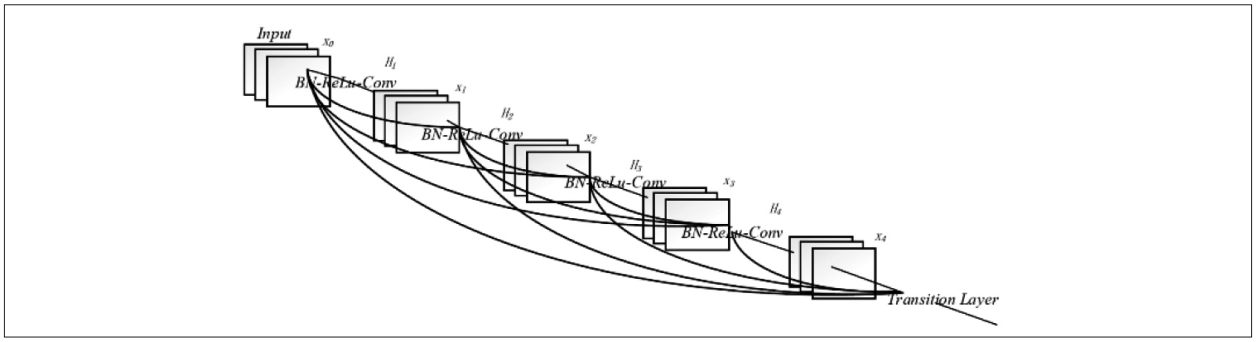
شکل ۳: نحوه انتخاب گره بعدی در قدم‌زنی سودار

از گره t که قبلاً دیده شده است در دو مرحله بعدی کم می‌شود مگر این‌که گره بعدی در گام دیگر همسایه‌ای نداشته باشد. پارامتر q جستجو بین گره‌های رو به داخل و رو به خارج را متمایز می‌کند. اگر $q > 1$ باشد، گام تصادفی به گره‌های نزدیک t متمایل است و در غیر این صورت در گام بعدی از گره t دور می‌شود.

شکل ۳ نقش پارامترهای p و q در محاسبه مقدار سوگیری را نشان می‌دهد. در این شکل از گره t به گره v گذر انجام شده و باید از گره v گام بعدی شروع شود. یکی از گره‌های x_1, x_2, x_3 یا t باید به‌عنوان گره مقصد انتخاب شود. مقدار سوگیری مرتبط با هر کدام روی پیوند ارتباطی‌شان با گره v نوشته شده است. حال بر اساس رابطهٔ (۲) مقادیر احتمالی $\pi_{vx}, x \in \{x_1, x_2, x_3, t\}$ محاسبه می‌شود. با استفاده از این مقادیر احتمالی و با استفاده از رابطهٔ (۱) گره مقصد به تصادف انتخاب می‌شود.

۲-۲- بازنمایی Node2vec استاندارد

روش Node2vec یک روش بازنمایی شبکه است که از قدم‌زنی سوگیری‌شده مرتبه دوم α استفاده می‌کند. فرض کنید با انجام این قدم‌زنی، مجموعه گره‌های شبیه به گره u، مجموعه $N_R(u)$ به‌وجود آمده است (R در $N_R(u)$ به راهبرد قدم‌زنی اشاره می‌کند). حال مسئله اصلی آن است که با فرض داشتن نشانندهٔ گره u بتوان به نشاننده‌هایی برای گره‌های مجموعه $N_R(u)$ رسید که بسیار نزدیک به نشانندهٔ گره u باشد. این موضوع را با معادله بهینه‌سازی رابطه(٤) نشان می‌دهیم. این رابطه به دنبال تابع ENC خاصی می‌گردد که احتمال رسیدن از گره u به گره‌های



شکل ۴: شبکه عصبی هم‌آمیختی متراکم متصل با نرخ رشد برابر ۳ و ۵ بلوک متراکم

شـــبیه به آن، $N_R(u)$، را بیشـــینه کند. در ایـــن رابطه z_u نشانندهٔ گره u است.

$$\max_{ENC} \sum_{u \in V} \log P(N_R(u)|z_u) \qquad (۴)$$

بر اساس رابطه فوق می‌توان یک تابع زیان تعریف کرد که در رابطه (۵) نشان داده شده است.

$$L = \sum_{u \in V} \sum_{v \in N_R(u)} -\log P(v|z_u) \qquad (۵)$$

الگوریتم بازنمایی Node2vec استاندارد با کمینه‌سازی تابع زیان فوق به نشـــانندهٔ مربوط به هر گره دســـت پیدا می‌کند. برای اطلاعات بیشتر به مرجع [۷] اشاره کنید.

۲-۳- شبکه عصبی هم‌آمیختی متراکم متصل

شبکه عصبی باقیمانده[6] نوعی شبکه عصبی هم‌آمیختی می‌باشـــد که با استفاده از اتصالات میانبر توانست مشکل محوشـــدگی گرادیان را حل کنـــد [۱۲]، امّا به دلیل محدود بودن اتصالات میان لایه‌ها جریان اطلاعات به خوبی انتقال پیـــدا نمی‌کند. به منظور انتقال بیشـــتر جریان اطلاعات در شبکه، یک مدل شـــبکه عصبی هم‌آمیختی متراکم پیشنهاد شـــد [۱۳]. در واقع، شـــبکه عصبی هم‌آمیختی متراکم یک مورد خاص از شبکه باقیمانده است، که ایده اصلی آن پرش اتصال است؛ برخی از ورودی‌ها بدون انتخاب وارد لایه‌ها می‌شـــوند (شـــکل ۵). اتصال میان لایه‌ها مبتنی بر پارامتر است که توسط کاربر در شبکه تنظیم می‌شود. در مقایسه با شـــبکه باقیمانده، که تنها یک «اتصال میانبر»[7] را معرفی می‌کند، شبکه عصبی هم‌آمیختی متراکم «اتصالات میانبر» بیشتری را نشان می‌دهد، همان‌طور که در شکل (٤) نشان داده شده است. این اطلاعات مرجع پردازش نمی‌شود بلکه مستقیماً انتقال می‌یابند.

معماری شبکه عصبی هم‌آمیختی متراکم به این‌گونه است که دارای چندین بلوک می‌باشـــد که اتصال میانبر متراکمی بیـــن آن بلوک‌هـــا وجود دارد کـــه در شـــکل (٤) به نمایش گذاشته شده است. درون هر کدام از این بلوک‌ها چندین لایه شـــبکه هم‌آمیختی که در بین هر کدام از این بلوک‌ها یک لایه هم‌آمیختی وجود دارد که لایه‌ای بر روی خروجی بلوک قبلی و ورودی بلوک بعدی می‌باشد. یکی از پارامترهایی که توسط کاربر تعیین می‌شـــود، تعیین تعداد بلوک‌های شبکه عصبی هم‌آمیختی متراکم می‌باشد که در این مقاله تعداد بلوک‌ها ٥ می‌باشد. در شبکه‌های هم‌آمیختی متراکم میزان انتقال جریان داده‌ها نیز به‌وســـیله کاربر تعیین می‌شود. به پارامتری که تعیین کننده میزان انتقال جریان داده‌ها اســـت «نرخ رشد»[8] گویند، که در این مقاله نرخ رشد برابر ۳ می‌باشد. نرخ رشد ۳ یعنی چه تعداد از فیلترهای لایه هم‌آمیختی به هر بلوک به بلوک بعدی بدون تغییر ارسال شود.

۳- کارهای گذشته

مهندســـی ویژگی دارای اهمیت بسیار زیادی در حوزه یادگیری ماشینی دارد. مهندســـی ویژگی به الگوریتم‌های یادگیری ماشینی کمک می‌کند تا درک بهتری از ویژگی‌های فضای مسئله به‌دســـت بیاورند. آموزش مدل با داده‌های بهتـــر منجر به ایجاد مدلی دقیقتر خواهد شـــد که در زمان کمتری نیز آموزش دیده است[۱۵].

6- Residual neural networks
7- Shortcut Connection
8- Growth rate

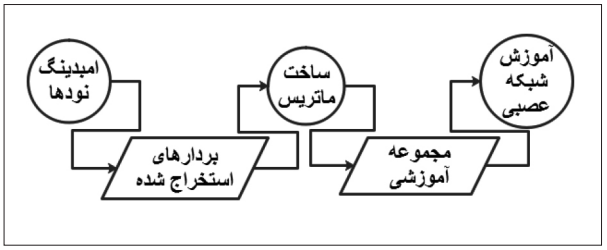
Skip-gram از جمله رویکردهای مهندسی ویژگی است که به منظور اســتخراج ویژگی از پرونده‌های متنی مورد استفاده قرار می‌گیرد[۱۶]. در این روش کلمات موجود در یک پرونده به نحوی بازنمایی می‌شــوند که کلمات مشابه، دارای شــباهت در فضای جدید باشند. فضای ایجاد شده جدید باعث ایجاد عملکــرد بهتر در الگوریتم‌های یادگیری ماشین می‌شود. رویکرد پیشنهادی الهام‌بخش رویکردهای مشابه در تحلیل شبکه‌های پیچیده شده است.

با توجه به کار [۱۶]، می‌توان کلمات را به‌عنوان گره‌ها و شباهت بین دو کلمه را معادل یال ارتباطی بین آن دو در نظر گرفت[۱۷]. در این کار با اســتفاده از ایده اولیه skip-gram توانســته‌اند روشی برای مهندســی ویژگی در شبکه‌های ارتباطی مختلف ارائه دهند. همان‌گونه که یک پرونده متنی حاوی چینش متوالی از کلمات اســت می‌توان گره‌های یک شــبکه را نیز به‌صورت متوالی در کنــار یکدیگر قرار داد. گره‌هایی که دارای یال ارتباطی باشــند دارای شباهت نیز هســتند. همچنین می‌توان از شباهت مبتنی بر همسایگی و یا شباهت مبتنی بر وزن استفاده کرد. مشکلی اصلی که به این دســت از کارها وارد است عدم وجود روشی مناسب برای چینش گره‌ها است.

روش [۷] به منظور رفع این مشکل ارائه شده است. این کار رویکردی با مفهوم نشاننده در فضای گراف پیشنهاد داده است که دیگر احتیاجی به چینش گره‌ها در یک فضای متنی نخواهد بود. این روش از قدم زنی سوگیری‌شــده به منظور پیمایش گره‌های گراف اســتفاده می کند. این روش توانســته اســت از رویکردهای متبنی بر skip-gram بهتر عمل کند. در این روش گره‌های همسایه یک گره، وزن‌دهی می‌شــود. مشــکل موجود در ایــن روش احتمــال وجود گره‌هایی با وزن یکسان خواهد بود.

۴- چارچوب پیشنهادی پیش‌بینی پیوند

چارچوب پیشنهادی DenseNet-LP در شکل ۵ نشان داده شده است. این چارچوب سه مرحله دارد.



شکل ۵: چارچوب پیشنهادی

- مرحله اول- استخراج نشاننده: در این مرحله با استفاده از یک الگوریتم یادگیری بازنمایی به ازای هر گره u نشاننده Z_u تولید می‌شود. قطعاً در این مرحله می‌توان از الگوریتم‌های متفاوتی اســتفاده نمود. در بخــش آزمایش‌ها، روش‌های متفاوتی مــورد آزمایش قرار گرفته اســت. در ادامه برای این بخش، الگوریتم بازنمایی جدیدی بر اساس Node2vec پیشنهاد می‌شود که Node2vec تعمیم‌یافته نام دارد.
- مرحله دوم-ســاخت ماتریس: در ایــن مرحله از روی نشــاننده‌های ورودی یک ماتریس ســاخته می‌شود. این ماتریس برای ورودی شــبکه عصبی که در مرحله ســوم وجود دارد الزامی است. در ادامه دو روش متفاوت ساخت ماتریس شرح داده خواهد شد.
- مرحله ســوم- اســتفاده از شــبکه عصبی هم‌آمیختی متراکم: در این مرحله، ماتریس ساخته شده مرحله قبلی به یک شبکه عصبی هم‌آمیختی داده می‌شود. این شبکه وظیفه رده‌بندی را انجام می‌دهد. برای یادگیری این شبکه به ازای هر جفت گره (u, v) ماتریس‌های ســاخته شده متناظر با آن‌ها به‌عنوان ورودی به شــبکه عصبی داده می‌شود. اگر بین این جفت گره، پیوندی در گراف وجود داشــته باشد، خروجــی رده اول و درغیرایــن صــورت رده دوم لحاظ می‌شود. بعد از آموزش شبکه عصبی کافی است ماتریس ساخته شده متناظر با یک جفت گره به شبکه داده شود تا وجود یا عدم‌وجود پیوند میان آن‌ها را پیش‌بینی کند.

در ادامه الگوریتم پیشــنهادی Node2vec تعمیم‌یافته، نحوه ســاخت ماتریــس در مرحله دوم و شــبکه عصبی موجود در مرحله سوم توضیح داده می‌شود. در انتها نیز شبه‌کد DenseNet-LP آورده شده است.

٤-١ Node2vec تعمیم‌یافته

با توجه به شـــکل ۳، برای قدم‌زدن تصادفی ســـه نوع گره مختلف در نظر گرفته می‌شـــود: گره منبع (مانند گره t در شـــکل ۳)، جاری (مانند گره V در شـــکل ۳) و مقصد (ماننـــد گره‌های x_1 و x_2 و x_3 در شـــکل ۳). در Node2vec تعمیم‌یافتـــه به ازای هر جفت گره منبع و گره جاری تمامی گره‌های مقصد ذخیره می‌شـــوند. بـــه ازای هر گره مقصد ذخیره شـــده یک عدد به‌عنوان احتمـــال آن گره نیز ذخیره می‌شود که بیانگر احتمال رفتن به آن گره است[۹].

الگوریتم Node2Vec استاندارد به ازای جفت گره‌های منبـــع و جاری و بعضی از گره‌های مقصد وزن احتمالاتی یکسان تولید می‌کند و این مشکل باعث می‌شود در مرحله قدم زدن تصادفی نتواند انتخاب مناســـبی از بین گره‌های محتمل داشته باشد و بنابراین گره‌های مقصد در این قدم زدن کاملاً تصادفی انتخاب خواهد شد. در این مقاله تمرکز بر رفع این مشـــکل با ارائه یک روش نمونه برداری جدید است. این روش بر حسب موقعیت گره مقصد، شباهت گره جـــاری و گره مقصد و همچنین اختلاف فاصله گره مقصد با سایر گره‌های گراف طراحی شـــده است. در این روش $\beta_x = \text{degree}_x$ برابـــر درجه گره x اســـت. در این روش درجه گره مقصـــد در انتخاب آن تاثیرگذار خواهد بود. به عبارت دیگر از بین گره‌ها با وزن یکسان، گرهی که درجهٔ بیشـــتری دارد و در عمل با گره‌های بیشتری در گراف در ارتباط اســـت؛ گـــره مهم‌تری خواهد بـــود. بنابراین از بین گره‌های مشابه در این روش، این گره انتخاب خواهد شد.

$$\alpha_{pq}(v,x) = \begin{cases} \frac{1}{p} * \beta_x & \text{if } d_{tx} = 0 \\ 1 * \beta_x & \text{if } d_{tx} = 1 \\ \frac{1}{q} * \beta_x & \text{if } d_{tx} = 2 \end{cases} \quad (۶)$$

$$\beta_x = \text{degree}(x)$$

رابطـــه (٦) الگوریتم Node2vec تعمیم یافته را نشـــان می‌دهد. با محاسبه مقدار α، محاسبه $\pi_{vx} = \alpha_{pq}(v,x).w_{vx}$ انجام می‌شود. در نهایت با استفاده از معادله (۱) قدم‌زدن تصادفی انجام می‌شود.

۹- به ازای هر v, t, x_i منحصر به فرد این عملیات تکرار می‌شود.

٤-٢- نحوه ساخت ماتریس مرحله دوم

در این مرحله فرض می‌شود که هر گره درون زیرگرافی اســـت که k گره در داخل آن قرار دارد؛ نزدیک‌ترین گره‌ها تشکیل این زیرگراف را می‌دهند. در بخش آزمایش‌ها مقدار مناســـب برای پارامتر k بررسی شده است. پس برای یک گره k همســـایه که با کم‌ترین گام بتوان به آنها دسترسی داشت را به‌عنوان زیرگراف در نظر بگیرید (در صورتی که تعداد همســـایه‌ها کمتر از k گره باشد در ساخت ماتریس سلول‌های متناظر تا k همســـایه را صفر در نظر بگیرید). برای ســـاخت ماتریس می‌توان از دو رویکرد زیر استفاده کرد(هر دو رویکرد مـــورد آزمایش قرار گفته و در بخش نتایج آمده است):

ســـطرهای ماتریس را نشانندهٔ گره‌ها تشکیل می‌دهند. طول بردار نشاننده را k در نظر بگیرید. با توجه به در نظر گرفتن k همســـایه برای هر گره مانند گره u، یک ماتریس $k \times k$ ساخته می‌شـــود. پس از ایجاد ماتریس، سطرهای ماتریس بر اســـاس شـــباهت با گره u مرتب می‌شوند؛ هر سطر یک بردار نشاننده است و برای یافتن شباهت با بردار نشـــانندهٔ u از ضرب داخلی دو بردار استفاده می‌شود. در واقع، هرچه یک همسایه به گره اصلی بیشتر شبیه باشد در سطر بالاتری از ماتریس قرار میگیرد.

در روش دوم، هماننـــد روش اول یـــک ماتریس $k \times k$ ایجاد می‌شـــود. با ایـــن تفاوت که ایـــن ماتریس، ماتریس همســـایگی زیرگراف استخراجی می باشـــد. پس از ایجاد ماتریـــس، مانند روش قبل ســـطرها را بر اســـاس معیار شباهت، مرتب خواهیم کرد.

٤-٣ شبکه عصبی مرحله سوم

بعد از ساختن ماتریس نمایشی برای هر گره، می‌توان وجود یا عدم‌وجـــود پیوند میان دو گره را پیش‌بینی کرد. برای این کار اســـتفاده از یک شبکه عصبی عمیق پیشنهاد می‌شـــود؛ در این مقاله یک شـــبکه عصبی densnet-101 با تابع بهینه‌ســـازی Adam استفاده شـــده است. ماتریس ســـاخته شـــده مربوط به دو گره، در قالب دســـته‌های ۳۲

عددی ورودی این شـــبکه هستند. برای آموزش این شبکه از اطلاعات موجود اســـتفاده می‌شود: اگر بین دو گره یال باشـــد، برچسب ۱ و در غیر این صورت برچسب صفر به شـــبکه عصبی آموزش داده می‌شـــود. بعد از آموزش، از این شـــبکه عصبی برای تعیین پیوند میان دو گره استفاده می‌شود.

٤-٤- شبه‌کد DenseNet-LP

در شکل ٦ شبه‌کد چارچوب پیشنهادی DenseNet-LP آورده شـــده است. متغیر SGs آرایه‌ای از کلید-مقدار است که به ازای هر گره (کلید)، لیســـت همســـایگانش را نشان می‌دهد (زیرگراف مرتبط با گره کلید). متغیر SSGs به ازای یک گره (کلید)، لیســـت همسایگان را به‌صورت مرتب شده بر اساس شباهت نشـــان می‌دهد. متغیر RVs نیز بازنمایی آرایه‌ای هر گره را نشـــان می‌دهد. برای به‌دســـت آوردن بازنمایی آرایه‌ای در خط ششم از تابع Node2vec استفاده شده اســـت. در انتهای شبه‌کد نیز، مدل شبکه عصبی ذکر شده در مرحله سوم آموزش داده شده است.

۵- نتایج

در این بخش شـــش آزمایش طراحی شـــده اســـت تا الگوریتـــم Node2vec تعمیم‌یافتـــه با ســـه الگوریتم دیگر Node2vec اســـتاندارد، Deepwalk و Graph2vec، براساس معیارهـــای accuracy ،AUC و زمان اجرا با هم مقایســـه شـــوند. برای پیاده‌ســـازی تمامی الگوریتم‌هـــا از کدهای گیت‌هاب استفاده شده است[10]. نتیجه اجرای هر کدام از این چهار الگوریتم به‌عنوان خروجی مرحله اول DenseNet-LP لحاظ می‌شـــود. بدین‌ترتیب امکان مقایســـه الگوریتم‌ها با DenseNet-LP فراهم می‌شود.

ادامه این قسمت از مقاله به‌صورت زیر است: در بخش بعـــدی معیار ارزیابی AUC و خطا توضیح داده می‌شـــود. سپس داده‌های اســـتفاده شده در آزمایش‌ها توضیح داده می‌شـــوند. بخش ۵-۳ ســـه آزمایش طراحی شده است تا تنظیمات گام دوم مدل DenseNet-LP مشـــخص شود. در بخش‌های بعـــدی (بخش‌های ٥-٤ تـــا ٥-٦) آزمایش‌های مختلفی با هدف مقایســـه الگوریتم‌های گفته شده طراحی شده است.

۵-۱- معیارهای ارزیابی

در این بخش دو معیار ارزیابی اســـتفاده شـــده در این قســـمت، یعنی AUC و خطای طبقه‌بندی در شبکه عصبی، مرور می‌شوند.

۵-۱-۱- معیار AUC

یکـــی از اصلی‌ترین معیارهای مقایســـه الگوریتم‌های پیش‌بینی پیوند AUC اســـت؛ احتمـــال آن که یک پیش‌بینی پیوند چقدر از حالت انتخاب تصادفی بهتر اســـت. به‌طور کلـــی، امتیاز AUC بین ۰,۵ تا ۱,۰ و هر چه بزرگ‌تر باشـــد مناسب‌تر اســـت. در AUC امتیازها براســـاس مشاهده‌ها به‌دست آمده از داده‌ها اســـت. تعریف رسمی‌در فرمول ۷ بیان شده است:

$$AUC = \frac{N' + 0.5N''}{N} \qquad (۷)$$

N تعداد مقایسه مستقل می‌باشد. N' تعداد دفعاتی است که امتیاز به‌دست آمده از حالت تصادفی بیشتر باشد. N'' تعداد دفعاتی است که امتیاز به‌دست آمده با حالت تصادفی برابر باشد.

۵-۱-۲-معیار خطای طبقه‌بندی شبکه عصبی

تابع خطا[11] یکی از توابعی است که در شبکه‌های عصبی نقش بسزایی ایفا می‌کند. این تابع نشان‌دهنده میزان خطای طبقه‌بندی در شـــبکه عصبی می‌باشد. هرگاه یک ماتریس همســـایگی به‌عنوان ورودی به شبکه DenseNet-LP برای آموزش داده شـــود، در لایه خروجی، پیش‌بینی نهایی را با برچسب موردنظر آن مقایسه می‌شود، که اختلاف بین این دو را خطا می‌نامیم. وجود خطا در شـــبکه DenseNet-LP به معنای آن اســـت که وزن‌ها باید اصلاح شـــوند. هرچه خطا کمتر باشد میزان اصلاح وزن‌ها کمتر می‌شود. خطای

10- [https://github.com/phanein/deepwalk], [https://github.com/aditya-grover/node2vec], [https://github.com/benedekrozemberczki/graph2vec]

11- Loss Function

DenseNet-LP's Pseudo-code

```
Precondition: K: The number of matrix dimensions,
ConectionStatus: is true if the two nodes are connected.
Nodes[] nodes;
SubGraphes <node, nighbornodes>[] SGs;
SortedSubGraphes <node, nighbornodes>[] SSGs;
RepresentationVectors<node, Vector>[] RVs;
for node in nodes:
 RVs[node] = node2vec(node);
 SGs[node] = extractsubgraph(node, K));
for node in SGs.nodes():
 matrix = AdjacencyMatrix(SGs[node]);
SSGs[node] = SortByLikelyHood(matrix, RVs.GetSubNodes(SGs[node]));
for node1 in SSGs:
 for node2 in SSGs:
  if(node1 == node2):
   countinue;
  model.train(node1, node2, conectionstatus);
```

شکل ۶: شبه‌کد چارچوب پیشنهادی DenseNet-LP

کمتر به معنای کیفیت بالای آموزش شبکه DenseNet-LP می‌باشد.

توابع بسیار زیادی جهت محاسبه خطا وجود دارند [۱۴] که با توجه به نیاز کاربران انتخاب می‌شوند. تابع خطای مورد استفاده در این مقاله «آنتروپی متقاطع»[12] می‌باشد. تعریف رسمی این تابع در فرمول ۸ بیان شده است.

$$H_{(o,c)} = -\sum_{c=1}^{M} y_{o,c} \log\left(p_{o,c}\right) \qquad (۸)$$

که در فرمول ۸، y نشان‌دهنده برچسب صحیح ورودی oo برای ردهٔ c می‌باشد. p درجه احتمال پیش بینی ورودی o از ردهٔ c است. M، تعداد رده‌ها می‌باشد.

۵-۲- داده‌ها و شرایط مرتبط در آزمایش‌ها

شبکه‌های مورد استفاده برای این قسمت در ادامه شرح داده شده و ویژگی‌های اصلی توپولوژیکی آن‌ها در جدول ۱ نشان داده شده است. $|V|$ تعداد گره‌ها، $|E|$ تعداد پیوند ها، $< K >$ میانگین درجه گره‌ها و CC ضریب خوشه‌بندی شبکه است. در ادامه توضیحاتی در مورد هر شبکه ارایه شده است.

- USAir: شبکه حمل و نقل هوایی ایالات متحده (USAir) که از ۳۳۲ گره و ۲۱۲۶ پیوند تشکیل شده است. گره‌ها نشان‌دهندهٔ فرودگاه‌ها هستند. اگر یک مسیر مستقیم بین دو فرودگاه برقرار باشد، بین دو گره متناظر پیوند برقرار است.
- C.elegans: این شبکه در واقع شبکه عصبی کرم‌هایی به نام Caenorhabditis elegans (C.elegan) است. این شبکه دارای ۲۹۶ گره است که هر یک نشان‌دهندهٔ یک نورون است. نورون‌ها دارای ۲۱۵۱ پیوند می‌باشد؛ اگر حداقل یک همایه[13] یا محل اتصال بین آن‌ها وجود داشته باشد، دو گره با هم در ارتباط هستند.
- Jazz: این شبکه، همکاری بین نوازندگان جاز را نشان می‌دهد. هر گره یک نوازنده جاز است و یک پیوند بیانگر همکاری دو نوازنده در یک گروه است.
- INF: نشان دهنده ارتباط بین کاربران یک نرم افزار

12- Cross Entropy

13- synspse

جدول ۱: مشخصات مجموعهٔ داده‌ها

Datasets	|V|	|E|	< K >	< CC >
USAir	۳۳۲	۲۱۲۶	۱۲,۸۱۰	۰,۷۴۹
C.elegans	۲۹۶	۲۱۵۱	۲۰	۰,۲۴۵۲
Jazz	۱۹۸	۲۷۴۷	۲۷	۰,۶۱۷
INF	۴۱۰	۲۷۶۵	۱۳,۴۹	۰,۴۶

تمـــاس تصویری اســـت. اگر یـــک کاربر، کاربـــر دیگر را به‌عنوان مخاطب در لیست مخاطبان خود داشته باشد بین آن دو کاربر پیوند وجود دارد.

بـــرای انجام آزمایش‌ها، شـــرایط زیر در اســـتفاده از داده‌ها لحاظ شده است:

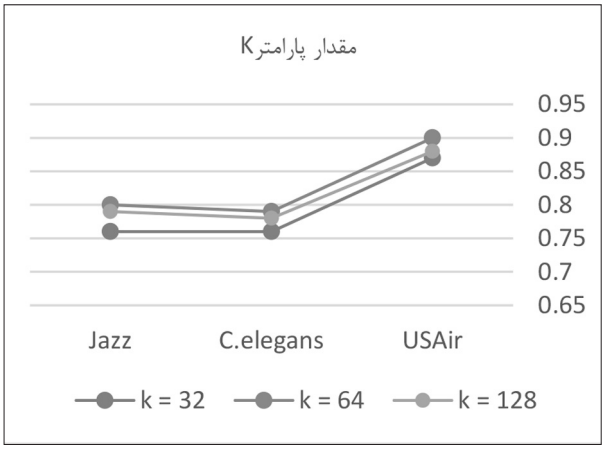
- مجموعـــه داده به‌صـــورت تصادفی و مســـتقل به یک مجموعه آموزشـــی (۹۰٪) و مجموعه آزمون (۱۰٪) تقسیم شدند. اطمینان از نبود اتصال بین شبکه مجموعه آموزش و مجموعه آزمون تضمین شده است.
- فراوانی جفت گره‌های بدون یال نسبت به جفت‌گره‌های با یال بیشـــتر است. برای جلوگیری از سوگیری وزن‌های شبکه عصبی به سمت یال نداشتن، نمونه‌برداری برای هر بار آزمایش به گونه‌ای است که تعداد جفت گره‌های بدون یـــال با تعداد جفت گره‌های با یال برابر اســـت. فرآیند این نمونه‌برداری کاملاً تصادفی اســـت. البتـــه نتایج آزمایش، متوسط تکرار ده باره این فرایند است.
- عمق اســـتخراج زیرگراف در ایـــن مقاله $h = 3$ تعیین شـــده است. پارامتر h تعداد فاصله گام‌ها از هر گره گراف می‌باشـــد. یعنی هر گره گراف تا همسایگی سوم آن از هر پیوند را شامل می‌شود.

۵-۳- تنظیمات گام دوم DenseNet-L

در این بخش سه آزمایش در دو زیربخش طراحی شده اســـت تا تنظیمات مورد نیـــاز در گام دوم DenseNet-LP به‌دســـت آید. این گام دو قسمت دارد: یکی تنظیم پارامتر k و دیگری انتخاب نحوه ســـاخت ماتریس. در ادامه، این دو زیربخش ارایه خواهد شد.

۵-۳-۱- تعیین اندازه ماتریس

تعداد ســـطر و ستون ماتریس مورد استفاده در بخش دوم DenseNet-LP از پارامترهای مهم چارچوب پیشنهادی است. در این مقاله فرض شده است که این ماتریس مربعی $k \times k$ اســـت. بنابراین مقدار مناســـب بـــرای پارامتر k از میان ســـه مقدار $\{32, 64, 128\}$ در آزمایش این زیربخش مشخص می‌شود.



شکل ۷: تعیین بهترین پارامتر برای k

نتایج آزمایش در شـــکل (۷) نمایش داده شـــده است. همان‌گونه که در شـــکل مشـــخص اســـت مقدار $k = 64$ بهترین AUC را برای تمامی مجموعه داده‌ها دارد. بنابراین در تمامـــی آزمایش‌های بعدی این مقـــدار برای پارامتر k استفاده شده است.

۵-۳-۲- انتخاب نحوه ساخت ماتریس

در ایـــن بخـــش دو آزمایـــش متفاوت طراحی شـــده اســـت؛ نحوه ســـاخت ماتریس (گام دوم DenseNet-LP) برای الگوریتم‌های Node2vec تعمیـــم یافته و Node2vec استاندارد، مورد بررسی قرار گرفته است. همان‌گونه که در قسمت ٤-٢ ذکر شد برای ساخت ماتریس $k \times k$ دو روش پیشنهاد شد: یکی ســـاخت ماتریس مبتنی بر همسایگی و دیگری ساخت ماتریس مبتنی بر بردارهای نشاننده؛ مقدار $k = 64$ بر اساس آزمایش قسمت ۵-۳-۱، برای این قسمت هم لحاظ شده است. بر اســـاس نتایج این دو آزمایش، در ادامه روش ســـاخت ماتریس بر اساس همسایگی استفاده خواهد شد.

آزمایش اول: در این آزمایش دو نحوه ساخت از طریق

جدول ۲: نتایج مقایسه الگوریتم‌های پیشنهادی ساخت ماتریس (گام دوم DenseNet-LP)

مجموعه داده	Node2vec استاندارد				Node2vec-تعمیم‌یافته			
	ساخت ماتریس مبتنی بر همسایگی		ساخت ماتریس مبتنی بر بردارهای نشاننده		ساخت ماتریس مبتنی بر همسایگی		ساخت ماتریس مبتنی بر بردارهای نشاننده	
	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy
USAir	0.88	0.89	0.85	0.85	0.90	0.92	0.86	0.88
C.elegans	0.73	0.81	0.74	0.75	0.79	0.84	0.81	0.8
Jazz	0.77	0.75	0.70	0.73	0.80	0.75	0.79	0.83
INF	0.80	0.85	0.76	0.78	0.85	0.89	0.82	0.82

معیارهای AUC و Accuracy با هم مقایسه شده‌اند. نتایج این آزمایش در جدول ۲ آمده است. این جدول برای هر کدام از الگوریتم‌های Node2vec استاندارد و تعمیم‌یافته نتایج را گزارش کرده است. هر سطر معیارهای AUC و Accuracy را به ازای یکی از مجموعه داده‌ها گزارش کرده است. بر این اساس، روش ساخت ماتریس بر اساس همسایگی شرایط بهتری دارد.

آزمایش دوم: این آزمایش، خطای طبقه‌بندی شبکه عصبی را به ازای چهار روش مختلف مقایسه کرده است؛ برای این آزمایش، دو الگوریتم Node2vec استاندارد و Node2vec تعمیم‌یافته با در نظر گرفتن دو روش پیشنهادی ساخت ماتریس چهار روش را ایجاد کرده است: Node-2vec استاندارد همسایگی، Node2vec استاندارد نشاننده، Node2vec تعمیم‌یافته همسایگی و Node2vec تعمیم‌یافته نشاننده. در روش‌هایی با اسم همسایگی، ماتریس مرحله دوم بر اساس همسایگی ساخته می‌شود و در روش‌هایی با اسم نشاننده، ماتریس بر اساس نشاندهٔ گره‌ها ساخته می‌شود. شکل (۸) خطای شبکه عصبی آموزش داده شده برای این چهار روش را نشان می‌دهد. این شکل دو موضوع را نشان می‌دهد: الف) ماتریس ساخته شده بر اساس همسایگی در دو الگوریتم کاراتر است و ب) در تمامی حالات الگوریتم Node2vec تعمیم‌یافته خطای کمتری دارد.

همان‌طور که دو آزمایش انجام شده نشان داد ماتریس‌های ساخته شده به کمک همسایگی دارای دقت بیشتر و خطای کمتر بوده‌اند به همین خاطر در ادامه کار از این ماتریس‌ها برای آموزش شبکه عصبی استفاده می‌شود.

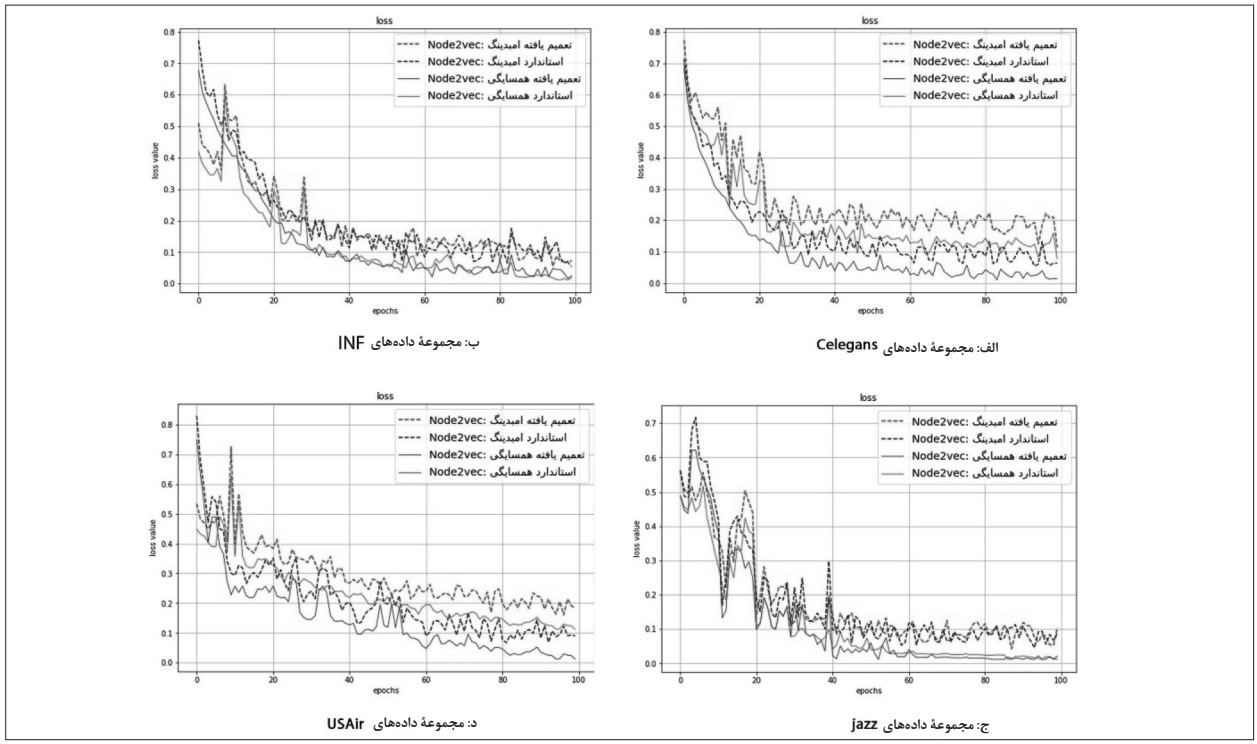
۵-٤- مقایسه کارآیی

در این بخش آزمایشی طراحی شده است که در آن کارایی الگوریتم‌های مختلف Node2vec تعمیم‌یافته، Node2vec استاندارد، DeepWalk و Graph2vec بر اساس معیارهای AUC و Accuracy با هم مقایسه می‌شوند. برای انجام مقایسه از DenseNet-LP استفاده شده است: گام اول این مدل با استفاده از هر کدام از این الگوریتم‌ها انجام می‌شود. برای گام دوم تمامی الگوریتم‌ها، $k = 64$ و ساخت ماتریس بر اساس ماتریس همسایگی استفاده شده است.

نتایج آزمایش در جدول ۳ آمده است. الگوریتم پیشنهادی Node2vec تعمیم‌یافته از Node2vec استاندارد و DeepWalk عملکرد بهتری دارد. مقایسه الگوریتم پیشنهادی با Graph2vec نشان می‌دهد که در برخی موارد عملکرد بهتری داشته است. امّا آیا زمان اجرای این الگوریتم‌ها با هم متفاوت است؟

۵-۵- آزمایش برابری و بهبود زمان اجرا

در این بخش آزمونی جهت مقایسه زمان آموزش شبکه عصبی هم‌آمیختی متراکم با الگوریتم Node2vec تعمیم‌یافته و سه الگوریتم دیگر طراحی شده است. برای هر چهار الگوریتم، زمان اجرای شبکه عصبی DensNet-LP



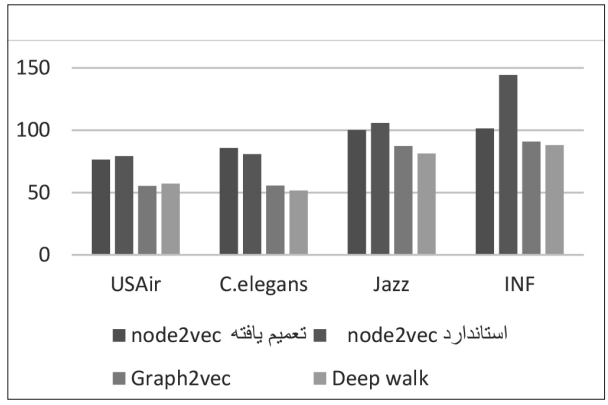
شکل ۸: میزان خطای شبکه عصبی با استفاده از نحوه ساخت ماتریس بر اساس همسایگی و بردار نشاننده

جدول ۳: مقایسه کارآیی الگوریتم‌های Node2vec تعمیم‌یافته، Node2vec استاندارد، DeepWalk و Graph2vec

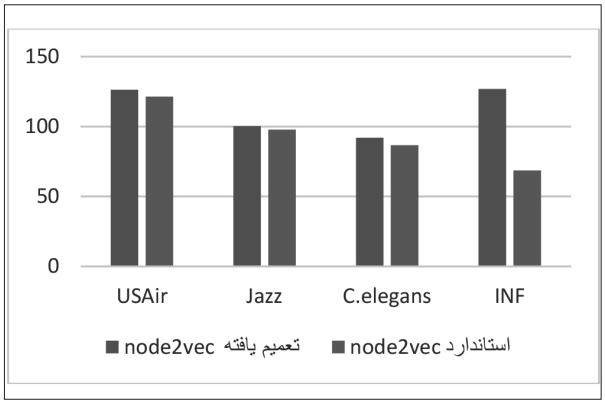
مجموعه داده	Node2vec استاندارد		Node2vec-تعمیم‌یافته		DeepWalk		Graph2vec	
	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy
USAir	0.88	0.89	0.90	0.92	0.84	0.82	0.85	0.88
C.elegans	0.73	0.81	0.79	0.84	0.70	0.69	0.80	0.80
Jazz	0.77	0.75	0.80	0.75	0.76	0.73	0.79	0.82
INF	0.80	0.85	0.85	0.89	0.76	0.75	0.73	0.79

برحسب ثانیه محاسبه می‌شود. برای هر مجموعه داده میانگین زمان مصرفی تمام epochها (مجموعا ۱۰۰ مورد) محاسبه و در شکل(۹) نمایش داده شده است. این آزمایش نشان می‌دهد که آموزش رویکرد پیشنهادی نسبت به Node2vec استاندارد از زمان تقریباً یکسانی برخوردار هستند. حتی این آزمایش نشان می‌دهد که رویکرد پیشنهادی زمان کمتری در بعضی از شبکه‌ها نسبت به رویکرد استاندارد دارد. این آزمایش بیان می‌کند که با وجود این موضوع که رویکرد پیشنهادی دارای AUC بهتری در آزمون کارایی است امّا همچنان دارای زمان آموزشی برابر با رویکرد استاندارد می‌باشد. امّا دو رویکرد دیگر دارای زمان اجرای کمتری هستند. ولی از لحاظ دقت از الگوریتم Node2vec تعمیم یافته ضعیف‌تر و در مواردی برابر عمل کردهاند.

امّا سوال اصلی آن است که چرا در شکل (۹) و در شبکه INF زمان الگوریتم Node2vec تعمیم‌یافته بسیار بهتر از Node2vec استاندارد است، در حالی‌که در سایر شبکه‌ها زمان این دو با هم برابر است؟ برای پاسخ به این سوال، تعداد میانگین یال‌های موجود در ماتریس‌های داده شده به شبکه عصبی (مرحله سوم DenseNet-LP)، برای این دو الگوریتم در شکل (۱۰) باهم مقایسه شده است. همان‌طور که از این شکل پیداست هرچه تعداد یال‌های موجود در



شکل ۹: زمان آموزش DenseNet-LP در شبکه‌های مختلف



شکل ۱۰: میانگین تعداد یالهای ماتریسها

ماتریس‌ها بیشـــتر باشد شبکه با ســـرعت بیشتری همگرا خواهد شد در نتیجه زمان کمتری برای آموزش نیاز خواهد داشـــت. روش Node2vec تعمیم‌یافته نسبت به Node2vec اســـتاندارد ماتریس‌هایی با اطلاعات بیشتر (یعنی یال‌های بیشـــتر) می‌سازد و این به حل مســـئله طبقه‌بندی (یا همان پیش‌بینی پیوند) در زمان کمتر کمک می‌کند.

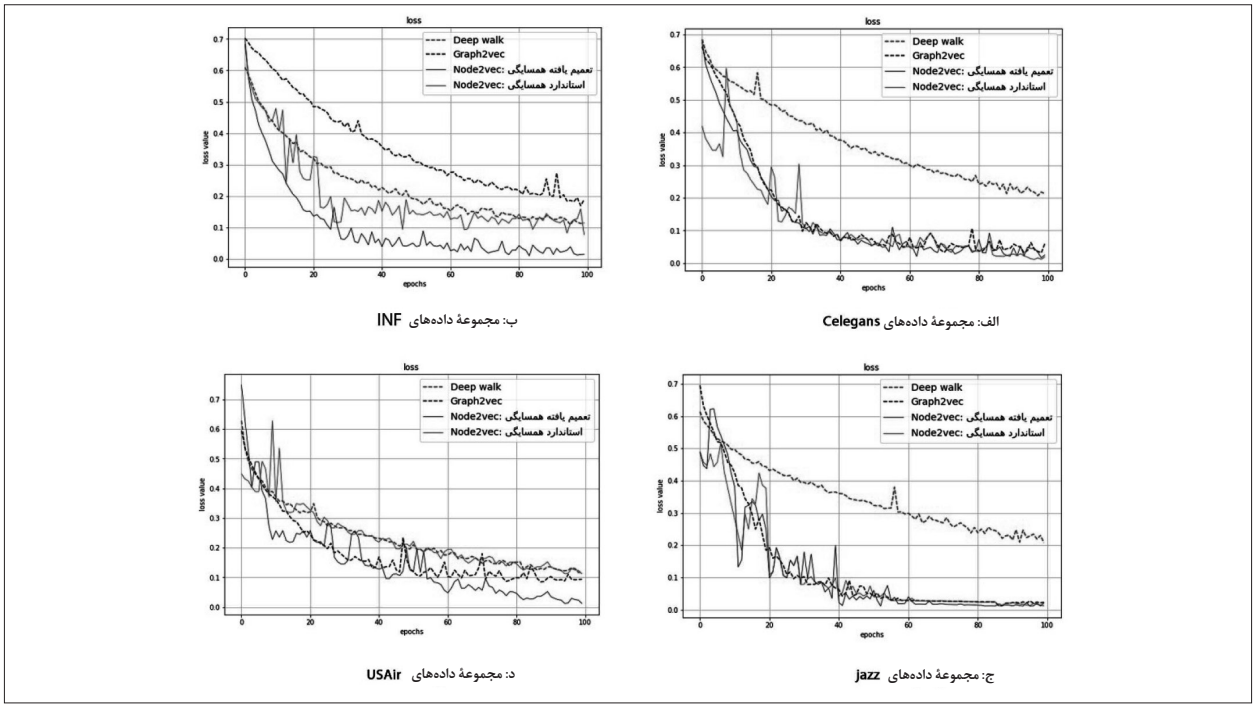
۵-۶- آزمایش کاهش خطا

در این بخش آزمونی جهت مقایسه کاهش خطای چهار الگوریتم طراحی شـــده اســـت. در الگوریتم پیشنهادی به علت ترکیب رویکردهای جستجو و نحوه چیدمان ماتریس اطلاعـــات گره، یافتن پیش‌بینی صحیح به وســـیله شـــبکه DenseNet-LP بهبود یافته اســـت. همین امر نشـــان دهنده خطای کمتر در طول آموزش مدل شـــبکه می‌باشد. در این آزمـــون روند کاهش خطای به‌دســـت‌آمده برای هر چهار الگوریتم گفته شده را محاسبه می‌کنیم.

شکل (۱۱) میزان خطای شبکه عصبی مربوط به چهار الگوریتم را در مجموعه داده‌های USAir, C.elegans, Jazz و INF نمایش داده اســـت. همان‌طور که در بالا اشاره شده است میزان خطای Node2vec بهبود یافته نسبت به سایر الگوریتم‌ها بهتر بوده اســـت که این امـــر به دلیل توجه به ساختار گراف در الگوریتم بهبود یافته است.

۶-نتیجه‌گیری

مســـئله پیش‌بینی پیوند یکی از مســـایل مهم در تحلیل شبکه‌های پیچیده است. از طرفی الگوریتم‌های پیش‌بینی بر اساس شـــباهت و یادگیری وجود دارند که هر کدام دارای مزایا و معایبی هستند. روش‌های پیش‌بینی براساس شباهت، توجهی به ویژگی‌های ســـاختاری شبکه ندارند. و از سوی دیگر حالت‌های غیرخطی را مورد بررســـی قرار نمی‌دهند. امّا روش‌های مبتنی بر یادگیری می‌توانند به‌صورت عمیق عمل کنند و ویژگی‌های غیرخطی و ساختاری را استخراج کنند. جهت پیش‌بینی پیوند به وســـیله الگوریتم‌های مبتنی بر یادگیری نیاز به اســـتخراج ویژگی‌های هر گره اســـت. مهندسی ویژگی روشی برای استخراج این ویژگی‌ها است. الگوریتم‌های بازنمایی امکان استخراج و انتقال ویژگی‌ها به یک فضای جدید اســـت. Node2vec از جمله الگوریتم‌های بازنمایی می‌باشد که ســـال‌های اخیر توسعه‌یافته و مورد توجه محققان بوده است. این الگوریتم بر اساس روش‌های جستجو، همســـایگی هر گره را به یک بردار تبدیل می‌کند. رویکرد پیشـــنهادی این مقاله در مرحله اول با استفاده از یک الگوریتم Node2vec تعمیم یافته بردارهای ویژگی هر گره استخراج می‌شود و سپس ماتریس (مجاورت یا مبتنی بر بردارهای بازنمایی)های مرتب‌شـــده‌ای ایجاد می‌شوند ســـپس این ماتریس‌هـــا به‌عنـــوان ورودی DenseNet-LP جهـــت پیش‌بینی پیوند اســـتفاده می‌شـــود. نتایج تجربی نشان می‌دهد که الگوریتم Node2vec تعمیم یافته بر روی بســـتر DenseNet-LP عملکرد بهتری نســـبت به الگوریتم Node2vec اســـتاندارد دارد. همچنین طبق نتایج به‌دســـت



شکل ۱۱: میزان خطای شبکه‌های عصبی به کمک چهار الگوریتم بازنمایی

آمده، مدل ســـاخته شـــده به کمک ماتریس‌های همسایگی دقیقتر از مدل ســـاخته شده به کمک ماتریس‌های نشاننده بوده و خطای کمتری نیز دارند.

مراجع

1. Tao, Z., & Zhang, J. (2013). Iterative MUSIC for highly correlated MEG source localization: A simulation study. Proceedings - 2013 4th World Congress on Software Engineering, WCSE 2013, 49(4), 217–220.
2. Yu, C., Zhao, X., An, L., & Lin, X. (2017). Similarity-based link prediction in social networks: A path and node combined approach. Journal of Information Science, 43(5), 683–695.
3. Sharma, P. K., Rathore, S., & Park, J. H. (2019). Multi-level learning-based modeling for link prediction and users' consumption preference in Online Social Networks. Future Generation Computer Systems, 93, 952–961.
4. Kunegis, J., & Lommatzsch, A. (2009). Learning spectral graph transformations for link prediction. Proceedings of the 26th International Conference on Machine Learning, ICML 2009, 561–568.
5. Zhang, D., Yin, J., Zhu, X., & Zhang, C. (2018). Network Representation Learning: A Survey. IEEE Transactions on Big Data, 6(1), 3–28.
6. Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining 701-710.
7. Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining 855-864.
8. Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., & Jaiswal, S. (2017). graph2vec: Learning distributed representations of graphs. arXiv preprint arXiv:1707.05005.
9. Schmidhuber, J. (2015). Deep Learning in neural networks: An overview. Neural Networks, 61, 85–117.
10. Zhang, Y., Tian, Y., Kong, Y., Zhong, B., & Fu, Y. (2020). Residual Dense Network for Image Restoration. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(9), 1–1.
11. Mehlhorn, Kurt, and Peter Sanders. "Graph traversal." Algorithms and Data Structures: The Basic Toolbox (2008): 175-189.
12. B, K. W., & Rutkowska, D. (2018). Linguistic Description of Color Images. Artificial Intelligence and Soft Computing, ICAISC 2017, 603–615.
13. Huang, G., Liu, Z., Pleiss, G., Van Der Maaten, L., & Weinberger, K. (2019). Convolutional Networks with Dense Connectivity. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1–1.
14. Kalambe, S., & Agnihotri, G. (2014). Loss minimization techniques used in distribution network: Bibliographical survey. Renewable and Sustainable Energy Reviews, 29, 184–200.
15. Bengio, Yoshua, Aaron Courville, and Pascal Vincent. (2013). Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence 35(8), 1798-1828.
16. Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
17. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In Proceedings of the 24th international conference on world wide web 1067-1077.