به نام خداوند بخشنده مهربان

شبکه‌های اجتماعی و اقتصادی
دانشکده‌ی مهندسی کامپیوتر

نام و نام خانوادگی: **شماره دانشجویی:**

مقاله Node2vec: Scalable Feature Learning for Networks

۱. هدف اصلی این مقاله چیست؟ هدف اصلی این مقاله ارائه روشی برای نمایش و یادگیری ویژگی های پیوسته برای راس های یک شبکه در یک فضای با بعد مشخص و تعبیه ی آن ها در آن فضا جهت بکار گیری در مسائلی مثل طبقه بندی راس و پیش بینی یال میباشد.

۲. روش های DFSو BFS هر یک چه معایب و مزایایی برای تولید دنباله ای از راس ها داشتند؟ آیا روش معرفی شده در این مقاله میتواند متناسب با هر مسئله عملکرد خود را به هر یک از متد های مذکور نزدیک کند؟ به طور دقیق توضیح دهید. روش bfs در فرایند جستجو راس هایی را دنبال میکند که نزدیک راس شروع باشند و برای مسائلی که وابسته به زیرگراف متناظر با حوالی یک راس هستند مناسب است و اگر فواصل طولانی در مسئله دخیل باشد این روش محدود است. روش dfs برخلاف bfs در فرایند جستجو با گذر زمان از راس شروع فاصله میگیرد و از آن دور میشود و برای مسائلی که مبتنی بر فاصله های طولانی از راس خاصی هستند مناسب است و برای مسائلی که مربوط به فاصله ی نزدیک یک راس باشند مناسب نیست. روش معرفی شده در این مقاله سعی کرده که متناسب با هر مسئله از روشی بینابینی استفاده کند که هم قابلیت تنظیم برای فاصله های نزدیک و هم برای فاصله های دور را داشته باشد:

Return parameter, *p*: Parameter p controls the likelihood of immediately revisiting a node in the walk. Setting it to a high value ($> \max(q, 1)$) ensures that we are less likely to sample an already-visited node in the following two steps (unless the next node in the walk had no other neighbor). This strategy encourages moderate exploration and avoids 2-hop redundancy in sampling. On the other hand, if p is low ($< \min(q, 1)$), it would lead the walk to backtrack a step (Figure 2) and this would keep the walk "local" close to the starting node u.

In-out parameter, *q*: Parameter q allows the search to differentiate between "inward" and "outward" nodes. Going back to Figure 2, if $q > 1$, the random walk is biased towards nodes close to node t. Such walks obtain a local view of the underlying graph with respect to the start node in the walk and approximate BFS behavior in the sense that our samples comprise of nodes within a small locality.

In contrast, if $q < 1$, the walk is more inclined to visit nodes which are further away from the node t. Such behavior is reflective of DFS which encourages outward exploration. However, an essential difference here is that we achieve DFS-like exploration within the random walk framework. Hence, the sampled nodes are not at strictly increasing distances from a given source node u, but in turn, we benefit from tractable preprocessing and superior sampling efficiency of random walks. Note that by setting $\pi_{v,x}$ to be a function of the proceeding node in the walk t, the random walks are 2^{nd} order Markovian.

۳. همانطور که از عنوان مقاله پیداست این روش scalable میباشد. به چه شکل این خاصیت را در گراف های بزرگ تضمین میکند؟ احتمال های عبور از هر یال را از قبل محاسبه میکند و هنگام تولید قدم زدن تصادفی از آن استفاده میکند و به شکل موازی تولید قدم زدن تصادفی و یادگیری تعبیه هارا انجام میدهد.

۴. آیا node2vec قابلیت پیاده سازی روی گراف های جهتدار و وزن دار را دارد؟ چگونه؟ بله، برای هر دو گروه گراف های وزن دار و جهتدار قابل استفاده است. برای گراف های جهتدار در زمان قدم زدن تصادفی جهت یالها را در نظر گرفته و در جهت هر یال حرکت میکنیم. برای گراف های وزن دار هم در محاسبه ی احتمال یالها وزن یال را هم دخیل میکنیم. مثلا برای هر یال وزن آن با احتمال آن نسبت مستقیم داشده باشد.

۵. اگر گرافی با انواع یالهای مختلف داشته باشیم(یال ها یکسان نباشند و لزوما یک نوع رابطه را مشخص نکنند). آیا روش معرفی شده میتواند در این نوع گراف ها اعمال شود؟ اگر بله توضیح دهید چرا؟ اگر نه سعی کنید با تغیراتی در مدل این قابلیت را به آن اضافه کنید.(توجه کنید که خاصیت symmetry باید حفظ شود) برای سادگی فرض کنید گراف بدون جهت است.

روش معرفی شده در این مقاله با فرض اینکه همه ی یال ها از یک نوع هستند فرایند تعبیه را انجام داده و این یعنی قابلیت تمیز دادن یالها را از هم در صورت متفاوت بودن یالها ندارد. به این ترتیب با اعمال این روش نمیتوان متفاوت بودن یالها را اعمال کرد. اما با تغییراتی میتوان این کار را انجام دارد. توجه کنید که باید روشی ارائه دهیم که علاوه بر تعبیه سازی راس ها، یال ها را هم تعبیه کند. مطابق مقاله اگر تابع f(u) راس u را تعبیه کند. تابع g(e) را تعبیه کننده یال نوع e در نظر میگیریم. به این ترتیب مشابه ایده ای که در مقاله به کار گرفته شده و از ضرب داخلی f(u) و f(v)برای محاسبه ی احتمال یال uv استفاده کرده، ما از ضرب داخلی وزن دار برای این منظور استفاده میکنیم و وزن ها را g(e) مشخص میکند. به طور مثال اگر f(u) و f(v) و g(e) به ترتیب برابر (1,1,1) و (1,2,3) و (1,0,2) باشند ضرب داخلی مذکور برابر 7 = 3*2 + 2 * 0 +1 خواهد شد. به این ترتیب خاصیت تقارنی حفظ میشود و برای هر یال هم تعبیه لحاظ شده و در فرایند یادگیری دخیل میشود.

۶. فرض کنید که گرافی دوبخشی و بدون جهت داریم. مثلا گرافی دوبخشی که یک بخش آن مربوط به فیلم ها و بخش دیگر کاربران باشند و اگر کاربری فیلمی را تماشا کرده باشد بین آن کاربر و فیلم مربوطه یالی وجود خواهد داشت. حال فرض کنید بخواهیم مسئله ی خوشه بندی را برای کاربران و فیلم ها در این گراف حل کنیم. (کاربران را خوشه بندی کرده و همین کار را برای فیلم ها هم انجام دهیم) متد node2vec چه چالش هایی در این مسئله دارد؟ برای حل آن راه حل پیشنهاد دهید.

روش node2vec اگر مستقیما روی گراف مسئله پیاده سازی شود با این محدودیت روبرو است که اولا بین خود فیلم ها و بین خود کاربران یالی نداریم و خوشه بندی درزیرگرافی که یالی ندارد کار پیچیده ای است. ثانیا فارغ از احتمال های محاسبه شده در قدم زدن تصادفی راس ها به شکل یکی در میان نوعشان تغییر خواهد کرد و همچنین طبقه بندی فیلم ها و کاربران همزمان انجام میشود و ویژگی های در نظر گرفته شده برای کاربران و فیلم ها یکسان میشود که منطقی نیست. یک ایده برای رفع این محدودیت ها این است که برای هر گروه کاربر و فیلم گرافی مجزا بسازیم و روی آن گراف ها متد node2vec را اعمال کنیم. به عنوان مثال یک ایده میتواند این باشد که گرافی وزندار با راس ها کاربران بسازیم و وزن یال بین دو راس uو v را برابر مجموع معکوس طول همه ی مسیر های ممکن بین آن دو راس در گراف اصلی بدهیم مثلا اگر بین u و v دو مسیر با طول ۴ و سه مسیر با طول ۲ وجود داشت وزن یال بین آن ها برابر ۲ خواهد شد. ویژگی این روش وزن دهی این است که هر چه طول مسیر های کوتاه بین دو راس بیشتر باشد وزن یال بین آن دو بیشتر خواهد شد و به این ترتیب احتمال قرارگیری آن دو راس در یک خوشه بیشتر میشود. همین کار را میتوان با فیلم ها انجام داد و به این ترتیب دو گراف مجزا برای کاربران و فیلم ها خواهیم داشت که میتوان با اعمال روش node2vec روی هر کدام مسئله را حل کرد.