به نام خداوند رنگین کمان

سری پنجم تکالیف درس گراف کاوی
پاسخ سوالات میان‌ترم

زهرا تبیانیان

سوال ۱

الف

راس	درجه	**closeness centrality**	**betweenness centrality**
1	2	0.14	0
2	4	0.16	8
3	2	0.11	0
4	1	0.1	0
5	2	0.125	4
6	1	0.083	0

بیشترین درجه، closeness centrality و betweenness centrality متعلق به راس ۲ می‌باشد. با توجه به اینکه هر سه این معیارها Importance-based هستند می‌توان این‌گونه تفسیر کرد که کاربر ۲ از دیگر کاربران مهم‌تر است. کمترین درجه برای رئوس ۴ و ۶ است. کمترین closeness centrality متعلق به راس ۶ و کمترین betweenness centrality متعلق به رئوس ۱، ۳، ۴ و ۶ است. به طور کلی می‌توان نتیجه گرفت که کاربر ۶ از همه کم‌اهمیت‌تر است.

درجه هر راس، تعداد همسایه‌های آن راس را بدون در نظر گرفتن اهمیت می‌سنجد. هرچه درجه هر راس بیشتر باشد آن راس مهم‌تر است.

معیار closeness centrality نزدیکی یک راس به مرکز را می‌سنجد و بیان می‌کند که یک راس مهم است اگر shortest path با طول کمی با بقیه رئوس داشته باشد.

به بیان betweenness centrality، یک راس مهم است اگر روی تعداد زیادی از shortest path های بین رئوس دیگر قرار گرفته باشد.

معیارهای Importance-based در پیش‌بینی راس‌های تاثیرگذار در گراف مفید هستند.

ب

معیار clustering coefficient اینکه چقدر همسایه‌های یک راس متصل(connected) هستند را می‌سنجد.

معیار Graphlet degree vector تعداد وقوع graphlet های متفاوت در گراف را می‌شمارد. توجه داریم که graphlet ها ساختار شبکه‌ی همسایه‌ی یک راس را توصیف می‌کنند.

هردوی این معیارها در دسته معیارهای structure-based قرار می‌گیرند. این معیارها برای پیش‌بینی نقشی که یک راس در گراف دارد مفید هستند.

محاسبه این معیارها در امتحان میان‌ترم آمده است!

ج

همان‌طور که گفته شد تاثیرگذارتر بودن با معیارهای Importance-based مشخص می‌شود. بنابر این طبق قسمت الف، کاربر ۲ تاثیرگذارترین است.

د

۱. distance-based feature :

- Shortest path distance between two nodes: این روش اهمیت راس‌هایی که درجه بالا دارند را درنظر نمی گیرد. در واقع قدرت اتصال را اندازه گیری نمی‌کند.

۲. local neighborhood overlap: تعداد همسایه‌های مشترک بین دو راس.

- common neighbors: $|N(v_1) \cap N(v_2)|$
- Jaccard's coefficient: $\frac{|N(v_1) \cap N(v_2)|}{|N(v_1) \cup N(v_2)|}$
- Adamic adar index:

$$\sum_{u \in N(v_1) \cap N(v_2)} \frac{1}{\log(k_u)}$$

۳. Global neighborhood overlap

- Katz index: تعداد همه‌ی گشت‌های بین دو راس را می‌شمارد. می‌دانیم تعداد گشت‌های بین دو راس u و v به طول l از توان l ام ماتریس مجاورت به دست می‌آید.

Adamic adar index	Jaccard's coefficient	common neighbors	path shortest	یال
3.32	0.2	1	1	$(1,2)$
1.66	0.33	1	1	$(1,3)$
1.66	0.5	1	2	$(1,4)$
1.66	0.33	1	2	$(1,5)$
0	0	0	3	$(1,6)$
3.32	0.2	1	1	$(2,3)$
0	0	0	1	$(2,4)$
0	0	0	1	$(2,5)$
3.32	0.25	1	2	$(2,6)$
1.66	0.5	1	2	$(3,4)$
1.66	0.33	1	2	$(3,5)$
0	0	0	3	$(3,6)$
1.66	0.5	1	2	$(4,5)$
0	0	0	3	$(4,6)$
0	0	0	1	$(5,6)$

سوال ۲

الف

قرار است که برای راس‌ها embedding به دست آوریم به طوری که شباهت دو راس در گراف اصلی با شباهت embedding این دو راس در فضای embedding تناظر یک به یک داشته باشند. شباهت در فضای embedding با ضرب داخلی تعیین می شود. اما می‌خواهیم شباهت دو راس در گراف اصلی را با random walks مشخص می‌کنیم. در واقع ایده کلی از این قرار است:

$$\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_v \underset{\text{similar, equal, or approximates}}{\approx} \text{probability that } u \text{ and } v \text{ co-occur on a random walk over the graph}$$

ایده اصلی الگوریتم node2vec این است که برای یافتن embedding مورد نظر با شرایط گفته شده، از random walkهایی استفاده کنیم که trade-off ای بین دیدهای محلی و سراسری

شبکه برقرار باشد. وقتی داریم به طور تصادفی در گراف گشت می‌زنیم می‌توانیم به دو صورت فکر کنیم: ۱) می‌توانیم به صورت depth-first search عمل کنیم و به عمق برویم(که این گشت به ما دید سراسری می‌دهد). ۲) می‌توانیم به صورت breadth-first search عمل کنیم و تنها به طور محلی به همسایه‌های راس مورد نظر نگاه کنیم(که این گشت به ما دید محلی می دهد). در گشت node2vec دو پارامتر p و q را داریم که با احتمال $\frac{1}{q}$ به راس بعد می‌رویم و از راس مورد نظر دور می‌شویم و با احتمال $\frac{1}{p}$ به راس قبلی بازمی‌گردیم.(البته نمی‌توان لفظ احتمال را به کار برد زیرا جمعشان یک نمی‌شود! اما صرفا معیاری را برای ما تعیین می‌کنند تا تصمیم بگیریم به کدام راس برویم. برای اینکه واقعا احتمال را بدست آوریم باید مقادیر $\frac{1}{p}$ و $\frac{1}{q}$ را نرمالایز کنیم.). به پارامتر p پارامتر return و به پارامتر q پارامتر In-out می‌گوییم.

تفاوت این الگوریتم با DeepWalk در چگونگی تعریف مجموعه راس‌های همسایه و random است.

ب

فرض کنیم می‌خواهیم از راس ۳ شروع کنیم و یک قدم زدن تصادفی به طول ۲ داشته باشیم. پارامتر p را ۲۰۰ و پارامتر q را ۱ در نظر می‌گیریم. در قدم اول ابتدا از ۳ به ۲ می‌رویم. حال باید احتمال‌های random walk را محاسبه کنیم:

$$n = \frac{1}{p} + \frac{1}{q} + \frac{1}{q} + 1 = 3.005$$

احتمال بازگشت از ۲ به ۳:

$$\frac{1}{p} \times \frac{1}{n} = 0.001$$

احتمال DFS_احتمال رفتن از ۲ به ۵ و احتمال رفتن از ۲ به ۴:

$$\frac{1}{q} \times \frac{1}{n} = 0.332$$

احتمال BFS_احتمال رفتن از ۲ به ۱:

$$1 \times \frac{1}{n} = 0.332$$

حال با توجه به احتمالات انتخاب می‌کنیم که به کدام راس برویم. مثلا در اینجا به راس ۵ می‌رویم و بنابر این random walk به طول ۲ با شروع از ۳ به این صورت است: ۳ ← ۲ ← ۵

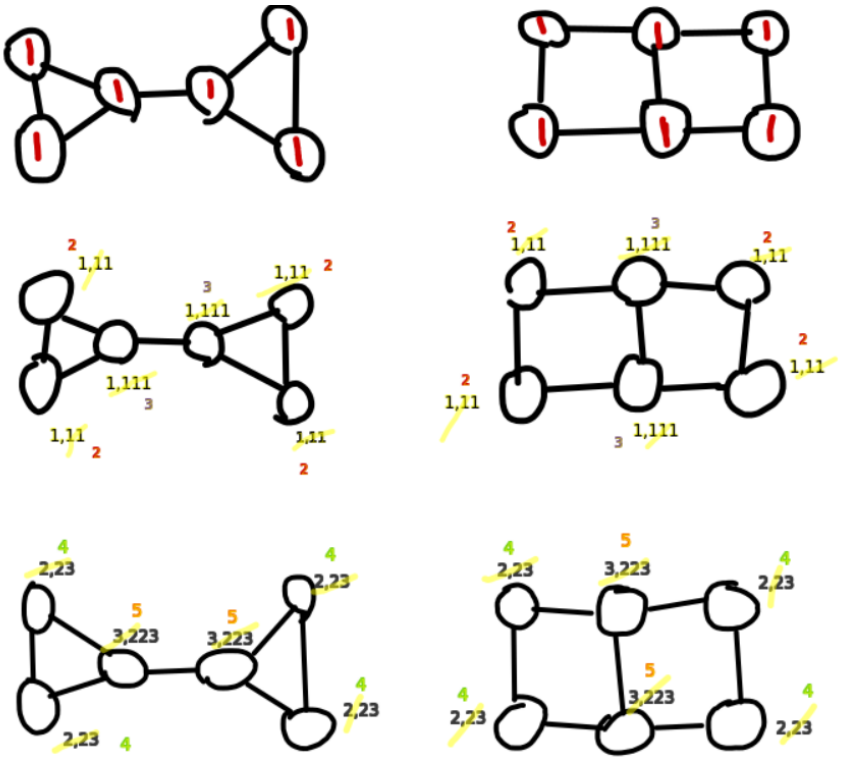
ج

الگوریتم node2vec هم ساختار محلی و هم سراسری را در نظر دارد در حالی که deepwalk تنها در task های محلی خوب کار می‌کند. از طرفی node2vec پیچیدگی محاسباتی بالایی دارد و در گراف‌های با سایز بالا خوب کار نمی‌کند.
به طور کلی اینکه از کدام استفاده کنیم بستگی به کاربرد دارد اما در مسائل کلاس‌بندی راس node2vec بهتر عمل می‌کند.

سوال ۳

الف

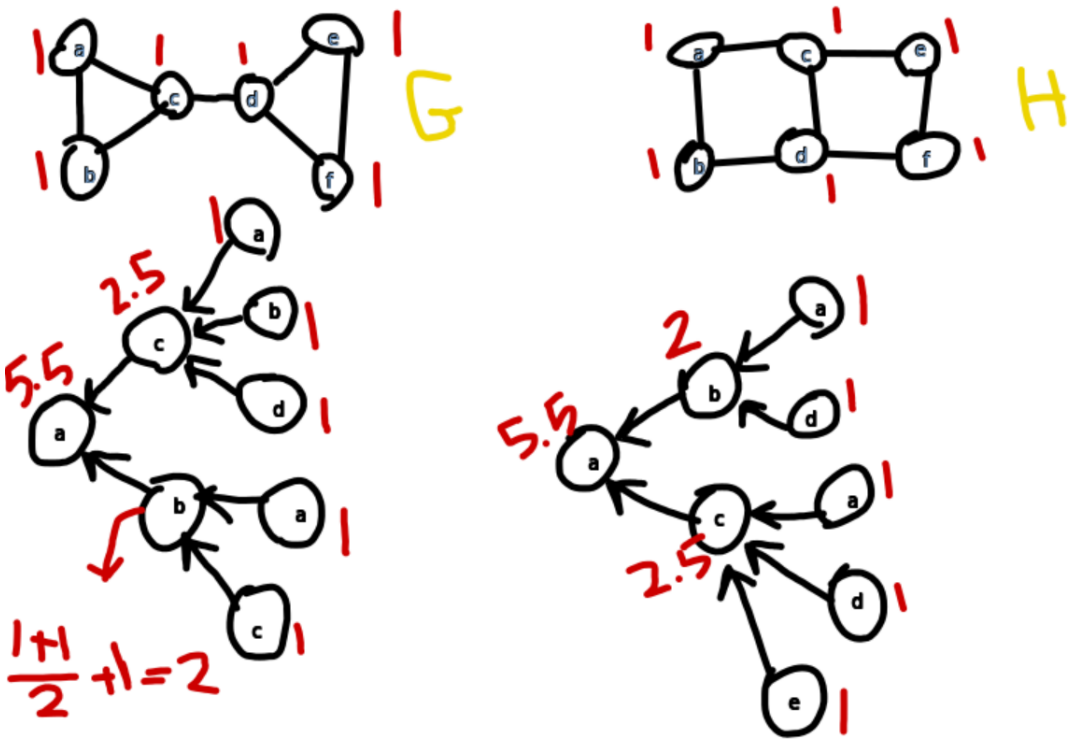
می‌خواهیم الگوریتم وایسفایلر_لیمن را اعمال کنیم. ابتدا رنگ اولیه ۱ را به همه راس‌های دو گراف نسبت می‌دهیم و الگوریتم را شروع می‌کنیم:



مشاهده می‌شود که الگوریتم در حلقه می‌افتد و مجموعه برچسب‌ها(رنگ‌ها) در دو گراف همسان است. در واقع ویژگی بدست آمده با این الگوریتم در دو گراف یکی خواهد بود بنا بر این، این الگوریتم قادر به تشخیص عدم یکریخت بودن این دو گراف نیست.

ب

مقدار اولیه راس‌ها را برای سادگی یک عدد و همگی را ۱ در نظر می‌گیریم:



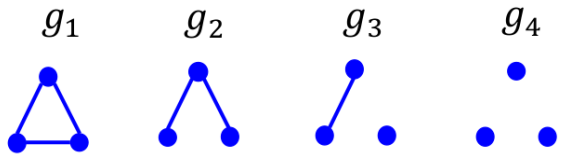
مشاهده می‌شود که embedding بدست آمده از computation graph برای دو راس به ظاهر متناظر a در دو گراف یکسان است. اگر computation graph را برای همه رئوس رسم کنیم می‌بینیم که یکی می‌شوند و embedding های بدست آمده یکی خواهد شد. بنابر این، این شبکه عصبی قادر به تشخیص عدم یکریختی این دو گراف نیست.

توجه داریم که با توجه به قضیه ای داریم: دو راس با computation graph معادل اند اگر

وتنها اگر با الگوریتم وایسفایلر_لیمن معادل باشند. از قسمت الف ثابت شد وایسفایلر_لیمن قادر به تشخیص عدم یکریختی این دو گراف نیست بنابر این با توجه به این قضیه با شبکه عصبی(computation graph) نیز قابل تشخیص نیست.

ج

از Graphlet Kernel استفاده می‌کنیم. با graphlet های سایز ۳ کار می‌کنیم:



یک بردار ۴ تایی که مولفه اول آن نشان‌دهنده تعداد g_1، مولفه دوم نشان دهنده تعداد g_2و... در هر گراف است، نشان دهنده‌ی ویژگی آن گراف است. این بردار را f می‌نامیم و داریم:

$$f_G = [2, 4, 12, 0]$$
$$f_H = [0, 10, 8, 0]$$

با توجه با اینکه این دو بردار باهم برابر نیستند می‌توان نتیجه گرفت که دو گراف G و H یکریخت نیستند.