Gráfbeágyazó (Graph embedding) algoritmusok időhatékonysága

Szakács Lili Kata

Önálló Projekt I., 2021/22 I. félév

1 Gráfbeágyazó (Graph embedding) algoritmusok

A gráfbeágyazó algoritmusok célkitűzése, hogy olyan alacsonydimenziós euklideszi vektorokat rendeljen hozzá gráfokhoz, melyekre teljesül, hogy a strukturálisan hasonló gráfok (például részfák, klikkek, fokszámeloszlás) reprezentációja egymáshoz közel legyen a beágyazott térben. Fontos elvárás továbbá, hogy a csúcsok permutációjára invariáns leképezést keresünk.

Két alapvető megközelítés létezik: a Laplace-mátrix spektrális tulajdonságainak statisztikus feldolgozása; strukturális tulajdonságok leképezése véletlen séták segítségével.

Ez a probléma több ponton is analóg a természetes nyelvfeldolgozás (NLP) területével. Több erre használt módszer alkalmaz beágyazó algoritmusokat, melyek a szavakhoz egy-egy vektort rendelnek olyan módon, hogy a szavak környezeteit vizsgálják, és hasonló jelentésű szavakhoz közeli vektorokat rendelnek. Gráfoknál a szavak szerepét a csúcsok veszik át, a szavak környezetei helyett pedig a csúcsokból indított random sétákkal nyert mintavételezést használjuk, majd a már bizonyított *Skipgram* modellel dolgozzuk fel.

2 Mérés

2.1 Kitűzött feladat

Fontos kérdés a gráfbeágyazó algoritmusokkal kapcsolatban a hatékonyságuk: ha valós hálózatok elemzésére szeretnénk használni a módszereket, akkor többmillió csúcsú gráfokon is belátható időn belül kell futniuk. Ennek ellenére a szakirodalom alig vizsgálja ezen eljárások hatékonyságát ezer csúcsnál nagyobb gráfokra.

A félév során a *KarateClub*[1] Python programcsomagban implementált gráf-beágyazó algoritmusok futásidejét vizsgáltam random generált gráfokon. Ez a programkönyvtár hatékony, state of the art módszereket is tartalmaz, ezért ez alapján tudunk következtetni arra, hogy nagyméretű valós gráfok beágyazása kivitelezhető-e ezekkel a módszerekkel.

2.2 Program

A mérés során skálafüggetlen input gráfokat használtunk annak érdekében, hogy a program jól becsülje nagy közösségi gráfok beágyazásának idejét. A *NetworkX* csomagban implementált *Barabási-Albert-modellt* használtuk, ennél a preferenciális kapcsolódást alkalmazó modellnél egy új csúcshoz $1-4$ új éllel is generáltunk $50-50$ gráfot különböző csúcsszám nagyságrendekre. Ezekre kipróbáltuk a következő, KarateClub csomagban implementált beágyazó eljárásokat, majd csúcsszám nagyságrendenként átlagos futásidőt számoltunk egy gráfra.

- **SF**[2]: Egyszerű baseline algoritmus, a gráf normalizált Laplace-mátrixának a k legkisebb nemnulla sajátértékét mint vektort rendeli egy gráfhoz.
- **IGE** *(Invariant Graph Embedding)*[3]: Három invariáns beágyazó algoritmus konkatenációja: az első az *SF*; a második random séták és egy csúcsokon definiált tulajdonságfüggvény hisztogramjából indul ki; a harmadik a Laplace-mátrix sajátvektorainak páronként vett skaláris szorzatainak hisztogramját használja.
- **NetLSD** *(Network Laplacian Spectral Descriptor)*[4]: Képzeljünk el a gráf egy csúcsába egy hőmennyiséget, majd vizsgáljuk ennek továbbterjedését, ezt az alábbi differenciálegyenlet írja le, illetve a megoldása a t időpillanatban:

$$\frac{\delta u_t}{\delta t} = -Lu_t \qquad H_t = e^{-tL} = \sum_{j=1}^{n} e^{-t\lambda_j}\phi_j\phi_j^T$$

  ahol L a gráf Laplace-mátrixa, λ_k-k a sajátértékei és ϕ_k-k a sajátvektorai. Ennek a mátrixnak a nyomát ($h_t = \sum e^{-t\lambda_j}$) véve különböző időpontokban kapjuk a gráfhoz a beágyazás vektorát.
- **FGSD** *(Family of Graph Spectral Distances)*[5]: Az alábbi metrika-családot definiálja csúcspárokon:

$$S_f(x,y) = \sum_{k=0}^{N-1} f(\lambda_k)(\phi_k(x) - \phi_k(y))^2$$

  ahol λ_k-k a gráf Laplace-mátrixának sajátértékei, ϕ_k-k a sajátvektorai, f pedig egy tetszőleges függvény, melyre $f(0) = 0$. Minden f megfeleltethető egy dimenziócsökkentő technikának, így minden gráfot be tudunk ágyazni az Euklideszi síkba egy *FGSD*, mint izometrikus mérték segítségével.
- **LDP** *(Local Degree Profile)*[6]: Minden csúcsnál kiszámolja a szomszédok fokszámának multihalmazára a minimumot, maximumot, átlagot és szórást, majd a csúcs fokával együtt alkot 5-dimenziós vektorokat csúcsonként. Mind az 5 mérőszámra hisztogrammal vagy empirikus eloszlással nyert vektorok konkatenációjával kapjuk a gráf leképezését.

- **Graph2Vec**[7]: Ebben a modellben úgy tekintünk a gráfokra és a csúcsok környezeteire, mint egy dokumentumra és szavaira: azt feltételezzük, hogy hasonlóan épülnek fel a szövegek szavakból, mint a gráfok a csúcsoknál gyökerező részfákból. Az elkészített random séták után a beágyazások tanulása a *doc2vec* dokumentumbeágyazó eljárás segítségével történik.
- **GL2Vec** *(Graph and Line graph to vector)*[8]: A *Graph2Vec* javítása azzal, hogy az élgráfot is feldolgozza, ezzel az élek címkéit és gazdagabb strukturális jellemzőket is figyelembe vesz az algoritmus.
- **GeoScattering**[9]: Definiálunk egy $x : V \mapsto R$ jelfüggvényt a gráf csúcsain, amelyre alkalmazni tudunk a random séták alapján számított *Wavelet* transzformációkat. Az így kapott vektorok momentum-jellegű mérőszámai megfelelő beágyazást fognak adni, amely már invariáns csúcspermutációra.
- **FeatherGraph**[10]: Ebben a módszerben adott hosszúságú random sétákon alapuló karakterisztikus függvényt definiálnak minden csúcshoz, ahol ezek jelentősen különböznek eltérő strukturális tulajdonságú csúcsokra. Ezek bizonyos (adott vagy tanult) pontokon való kiértékelésével kapjuk a csúcsok beágyazását, melyből mean poolinggal kapjuk az egész gráfhoz tartozó vektort.

2.3 Eredmények

A program futása instabil volt 10^4-nél nagyobb nagyságrendre, a *FeatherGraph, LDP, Graph2Vec* és *SF* bizonyultak elég robosztusnak, azokat is csak 10^5 nagyságrendig tudtuk kipróbálni. Az ábrán látható, hogy az eljárások futásideje hogyan függ a gráfok méretétől – nagyjából lineáris.

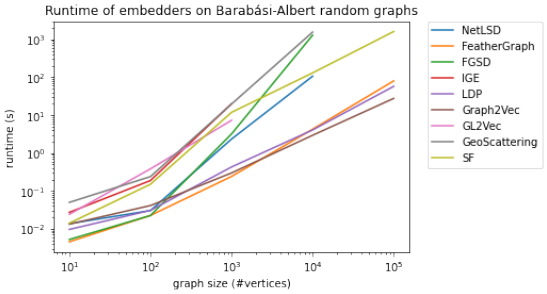

Figure 1: Gráfbeágyazási módszerek futási ideje

3 További célok

Az eddigi futtatásokkal sikerült megállapítani, hogy a fent említett *FeatherGraph, LDP* és *Graph2Vec* az a 3 eljárás a 9-ből, melyeket érdemes lehet majd használni nagy méretű hálózatokon (az *SF* is lineáris ugyan, de sokkal lassabb).

Még nem foglalkoztunk az egyes módszerek performanciájának elemzésével klasszifikációs feladatoknál, csak a futásidőket hasonlítottuk össze, így fontos lenne igazán nagy gráfokon vizsgálni a kiválasztott eljárások teljesítményét.

References

[1] B. Rozemberczki, O. Kiss, and R. Sarkar, "An API oriented open-source python framework for unsupervised learning on graphs," *CoRR*, vol. abs/2003.04819, 2020.

[2] N. de Lara and E. Pineau, "A simple baseline algorithm for graph classification," *CoRR*, vol. abs/1810.09155, 2018.

[3] A. Galland and M. Lelarge, "Invariant embedding for graph classification," in *ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data*, (Long Beach, United States), June 2019.

[4] A. Tsitsulin, D. Mottin, P. Karras, A. M. Bronstein, and E. Müller, "Netlsd: Hearing the shape of a graph," *CoRR*, vol. abs/1805.10712, 2018.

[5] S. Verma and Z.-L. Zhang, "Hunt for the unique, stable, sparse and fast feature learning on graphs," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[6] C. Cai and Y. Wang, "A simple yet effective baseline for non-attribute graph classification," *CoRR*, vol. abs/1811.03508, 2018.

[7] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," *CoRR*, vol. abs/1707.05005, 2017.

[8] H. Chen and H. Koga, "Gl2vec: Graph embedding enriched by line graphs with edge features," in *Neural Information Processing - 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12-15, 2019, Proceedings, Part III* (T. Gedeon, K. W. Wong, and M. Lee, eds.), vol. 11955 of *Lecture Notes in Computer Science*, pp. 3–14, Springer, 2019.

[9] F. Gao, G. Wolf, and M. Hirn, "Geometric scattering for graph data analysis," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 2122–2131, PMLR, 09–15 Jun 2019.

[10] B. Rozemberczki and R. Sarkar, "Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models," 2020.