Gráfbeágyazó (Graph embedding) algoritmusok performanciája

Szakács Lili Kata

Önálló Projekt II., 2021/22 II. 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.

A beágyazott gráfok már alkalmasak arra, hogy klaszterezési feladatok kényelmes és hatékonyan előfeldolgozott bemenetei legyenek – például hálózatok paramétereinek vagy jellegük előrejelzésére.

2. Mérés

2.1. Kitűzött feladat

Kulcskérdés a gráfbeágyazó algoritmusokkal kapcsolatban a hatékonyságuk "előfeldolgozásként" klaszterezési feladatoknál. Ennek gyakorlati haszna is lehet: ha valós hálózatok elemzésére szeretnénk használni az embedding módszereket, akkor reális elvárás, hogy a hálózat jellegét leíró tulajdonságok megkülönböztetésére alkalmasak legyenek. Ilyen például egy gráf fokszámeloszlása (pl. skálafüggetlen vagy normális), egyes gráfmodellekben a paraméterek, vagy maguk a modellek is akár.

A félév során a *NetworkX*[1] Python programcsomagban implementált random gráf generáló algoritmusok közül választottunk párat, melyeket utána a *KarateClub*[2] Python programcsomagban implementált embedding algoritmusokkal ágyaztunk be. A beágyazással kapott vektorokat néhány klasszifikációs feladathoz használtam fel.

2.2. Program

2.2.1. Gráf generáló modellek

Minden általunk generált gráf 1000 és 1100 közötti csúcsszámmal rendelkezett, és a beágyazó algoritmusok elvárásának megfelelően összefüggő volt. A mérés során reális, skálafüggetlen és más jellegű random gráf modelleket is használtunk, különböző paraméter beállításokkal, melyek az alábbiak voltak:

- **Barabási-Albert-modell (BA)**: preferenciális kapcsolódást alkalmazó modell, nálunk egy új csúcshoz 1-6 új éllel – *skálafüggetlen gráfok*
- **Duális Barabási-Albert-modell (DBA)**: preferenciális kapcsolódást alkalmazó modell, egy új csúcshoz p valószínűséggel egyfajta, $1-p$ másfajta új élszámmal (várható értéke ennek is nálunk 1 és 6 között) – *skálafüggetlen gráfok*
- **Holme & Kim-algoritmus (HK)**: preferenciális kapcsolódást alkalmazó modell, mely egy fix valószínűséggel összeköti az új csúcsot a szomszéd szomszédjával is, így háromszögeket alkotva – *skálafüggetlen gráfok*
- **Erdős-Rényi-modell (ER)**: adott csúcsszámú gráfban minden élt p valószínűséggel húzunk be (nálunk p $0,075$ és $0,3$ közötti) – *gráfok fokszámeloszlása binomiális, aszimptotikusan normális*
- **Random reguláris gráf (RR)**: Adott csúcsszámú és fokszámú reguláris gráfot generál az algoritmus, amely aszimptotikusan megfelel az egyenletesen véletlen választásnak ebből a gráfcsaládból (nálunk 4 és 12 közötti fokszámokkal) – *gráfok fokszámeloszlása egyenletes*

Egy méréshez minden modellből összesen nagyjából 90 gráfot generáltunk, azon belül adott paraméterűekből körülbelül $10 - 20$ darabot.

2.2.2. Karateclub – Embedding eljárások

A generált gráfokat eljárástól függően 100 és 500 közé eső dimenziójú valós térbe ágyaztuk be. Méréseim során a *Karakteclub* csomagban implemetált alábbi eljárásokat használtam alapértelmezett paraméter beállítások mellett:

- **SF**[3]: Egyszerű baseline algoritmus, a gráf normalizált Laplace-mátrixának a k legkisebb nemnulla sajátértékét mint vektort rendeli egy gráfhoz.
- **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.

  Mivel ez az eljárás nagyon lassú futásidővel gyenge eredményeket mutatott az első mérés során, a továbbiakhoz nem használtam.
- **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.

2.2.3. Klasszifikációs feladatok

Az embedding algoritmusok tesztelésére kétféle típusú predikciós feladatot találtunk ki: a beágyazott vektorok alapján az eredeti gráfot generáló modelleket, vagy egy azonos modell különböző paraméterek mellett generált példányait kellett elkülöníteni. A vizsgált klasszifikációs feladatok a következők voltak:

- **Barabási-Albert típusok**: a Barabási-Albert, duális Barabási-Albert és Holme-Kim (preferenciális kapcsolódást alkalmazó, skálafüggetlen) modellek megkülönböztetése

  3 kategória, bennük kb. 70 gráf a tanító halmazban és 20-22 gráf a teszt halmazban

- **Barabási-Albert paraméterei**: a Barabási-Albert modellben a paraméter (új csúcs bekapcsolódó éleinek a száma) meghatározása

  6 kategória, bennük kb. 10-15 gráf a tanító halmazban és 3-5 gráf a teszt halmazban

- **Duális Barabási-Albert-modell paraméterei**: a duális Barabási-Albert modellben a paraméter (az új csúcs bekapcsolódó éleinek várható értékének) meghatározása

  5 kategória, bennük kb. 14-17 gráf a tanító halmazban és 4-5 gráf a teszt halmazban

- **Barabási-Albert típusok paraméterei**: a Barabási-Albert és duális Barabási-Albert modellből kapott gráfok paramétereinek (az új csúcs bekapcsolódó éleinek várható értékének) meghatározása

  6 kategória, bennük kb. 25-35 gráf a tanító halmazban és 5-10 gráf a teszt halmazban

- **Különböző jellegű modellek**: a Barabási-Albert, az Erdős-Rényi és a random reguláris gráf modellek (más fokszámeloszlású, így egymástól jelentősen különböző tulajdonságokkal rendelkező gráfok) megkülönböztetése

  3 kategória, bennük kb. 70 gráf a tanító halmazban és 20-22 gráf a teszt halmazban

- **Minden modell**: mind az öt használt modell (BA, DBA, HK, ER, RR) megkülönböztetése

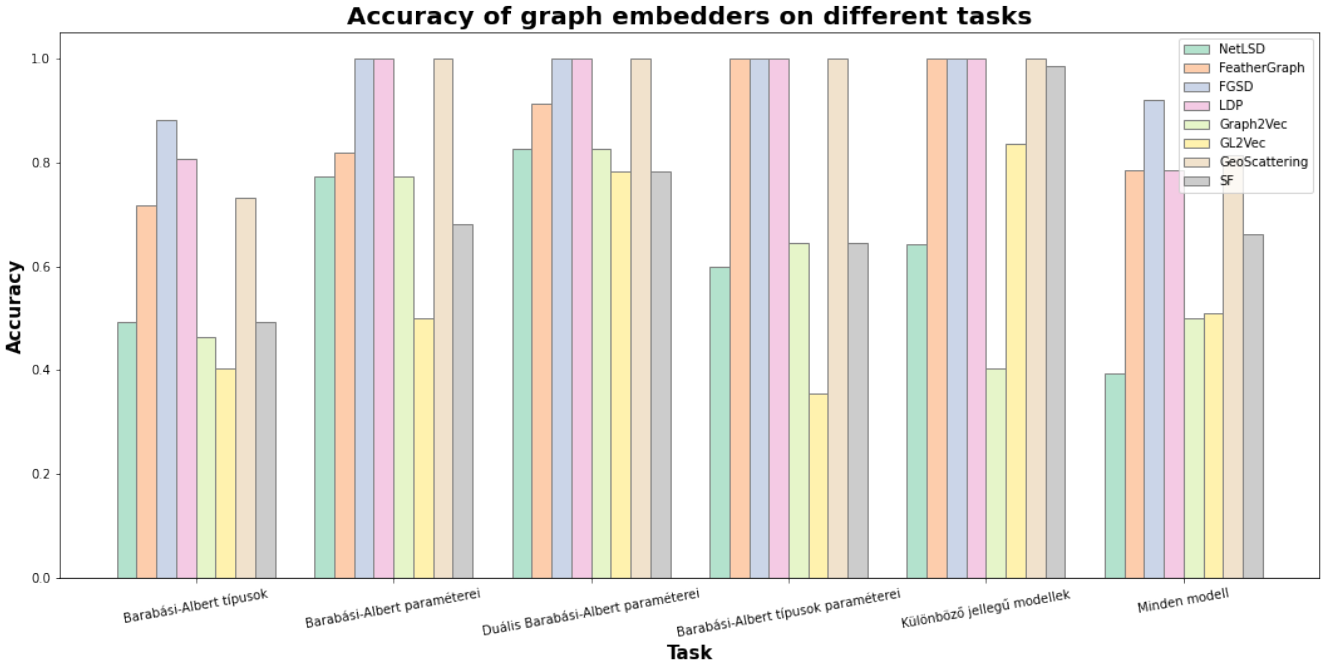
  5 kategória, bennük kb. 70 gráf a tanító halmazban és 20-22 gráf a teszt halmazban

A klasszifikációs feladatokhoz a (megfelelő gráfmodellekből kapott) beágyazott vektorok halmazát tanító és tesztelő adathalmazra osztottam (alapértelmezetten $0,75 - 0,25$ aránnyal), majd logisztikus regressziót alkalmaztam a *scikit-learn* [11] Python programcsomagból *Newton CG (Newton konjugált gradiens módszer)* optimalizálóval (a többi beállításnál az alapértelmezettet használtam).

2.3. Eredmények

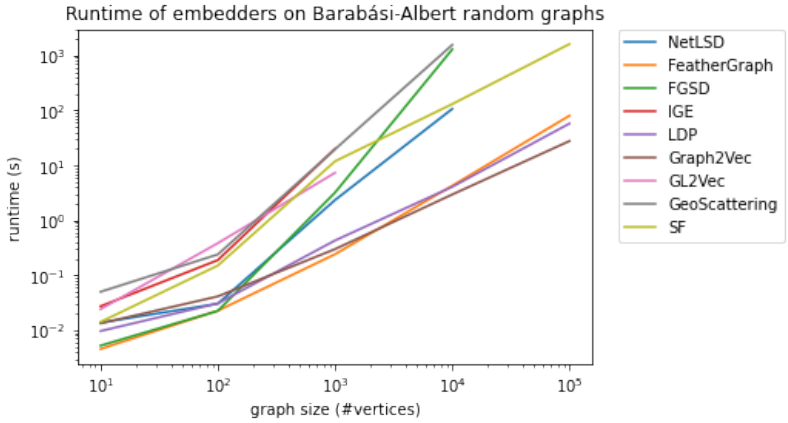
A fent ismertetett 6 feladaton az általam vizsgált 8 embedding algoritmus teljesítményét az első mérés alapján az 1. ábra foglalja össze.

Látható, hogy kiemelkedő teljesítményt nyújt a *FeatherGraph, FGSD, LDP* és *GeoScattering* eljárás – ezek közül különös meglepetést okozott az *LDP*, hiszen ez a legegyszerűbb és leggyorsabb eljárás mind közül.



1. ábra. Gráfbeágyazó módszerek performanciája különböző predikciós feladatokon

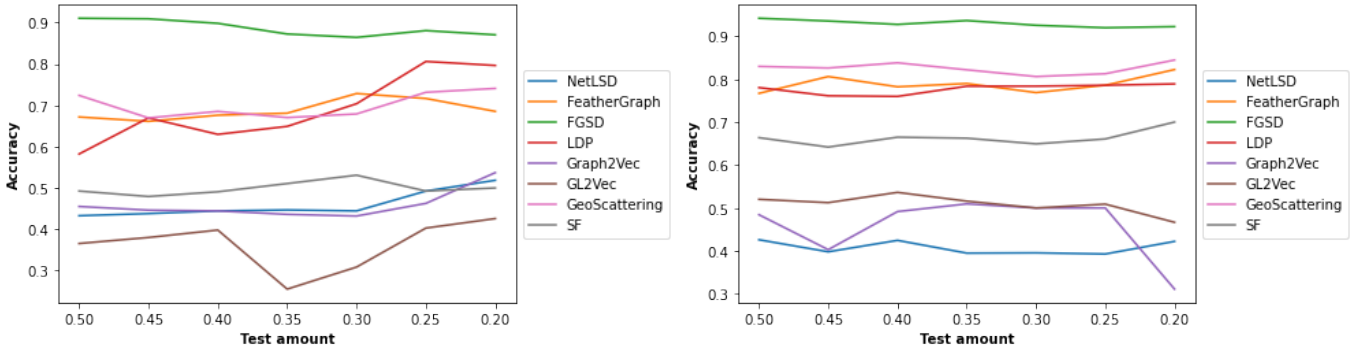
Az előző féléves futásidő mérések alapján az *FGSD* és a *GeoScattering* a leglassabb modellek közé tartozott, így ezek használata kompromisszumokkal jár.



2. ábra. Gráfbeágyazó algoritmusok futásideje – Projet I. mérés eredménye

2.3.1. Tanító adathalmaz mérete

Kísérleteztünk a tanító adathalmaz méretével, mennyire lehet levinni a tanító halmaz elemszámát komoly performanciaromlás nélkül. Nem látszott komoly eltérés az eredeti 25%-os teszt-arányhoz képest, a legjobb eljárások végig 1 közeli eredményt hoztak. Az első és az utolsó klasszifikációs feladat eredményei a teszt-arány függvényében:



(a) Barabási-Albert típusok

(b) Minden modell

3. ábra. Eljárások performanciájának alakulása két klaszterezési feladaton a teszt-halmaz aránya szerint

A kérdés valószínűleg érdekesebb lenne nagyobb méretű adathalmazon a gyengébb embedding eljárásokkal, és diverzebb (pl. jobban eltérő méretű), esetleg kicsit hibás adathalmazon a nagyon erős eljárásokkal.

Hivatkozások

[1] „Networkx package.” https://networkx.org/documentation/stable/reference/generators.html.

[2] 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.

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

[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.

[11] „scikit-learn package." https://scikit-learn.org/stable/index.html.