Raport științific și tehnic

privind implementarea proiectului în perioada august-noiembrie 2020
Proiect 287PED/2020 “Graphomaly”

Planul de realizare al proiectului Graphomaly prevede pentru anul 2020 următoarele activități:

1. Proiectare structură bibliotecă, format date.
2. Dezvoltare programe de generare și conversie de date.
3. Adaptare programe de detecție de comunități.
4. Implementare Python pentru biblioteca Dictionary Learning (DL) Matlab existentă.

Acest raport descrie rezultatele obținute în cadrul activităților, alte informații referitoare la desfășurarea proiectului, precum și concluzii.

0. Introducere

Proiectul Graphomaly are ca scop producerea unei biblioteci de programe pentru ***detecția anomaliilor în grafuri*** ce modelează tranzacții financiare; prin anomalii înțelegem în primul rând posibile tranzacții frauduloase cu caracter complex, cum ar fi spălarea de bani. Metodele avute în vedere țin în special de învățarea dicționarelor (Dictionary Learning - DL), acolo unde se situează competența noastră principală, dar vor cuprinde și alte tehnici de învățare automată.

În primele cinci luni ale proiectului, cele din anul 2020, ne-am propus activități prin care să creăm fundația pe care se va construi biblioteca și prin care membrii celor trei echipe participante (UPB, UB, Tremend) să eficientizeze dezvoltarea și testarea programelor ce vor realiza scopul principal, detecția de anomalii.

1. Structura bibliotecii și formatul datelor

Scopul în această etapă a fost stabilirea unor structuri de bază ale bibliotecii și a categoriilor de funcții, precum și un prim pas spre standardizarea formatelor datelor, cu atenție specială pentru cele de tip graf. Prezentăm aici informații generale asupra convențiilor adoptate.

1.1 Formatul datelor

Băncile și organizațiile financiare produc de obicei date în format tabelar, în care fiecare tranzacție reprezintă o linie din tabel, iar coloanele tabelului conțin informații despre părțile implicate în tranzacție, natura tranzacției (suma, valuta, etc.), precum și multe alte detalii (timp, operator, etc.), ajungându-se și până la sute de câmpuri. Nu există un format standard și, deși unele câmpuri sunt comune, nu există neapărat denumiri standardizate pentru toate câmpurile.

Structura de graf asociată este naturală:

- nodurile sunt conturi sau clienți
- arcele sunt tranzacțiile.

Pentru grafuri sunt posibile mai multe moduri de memorare:

- Format similar cu ideea de graf: obiecte conectate reprezentând noduri sau arce.
- Format tabelar, în care grafurile sunt memorate prin două tabele, unul de noduri, altul de arce, fiecare conținând atributele asociate.
- Matrice de adiacență, în care nodurile sunt etichete de linii și coloane, iar elementele nevide reprezintă arcele. Fiind vorba de o matrice rară, sunt posibile mai multe formate eficiente de memorare.

Figura 1 conține formatele selectate și relațiile dintre ele. Deoarece există deja biblioteci Python dedicate grafurilor și care cuprind diverse funcții generale de operare pe grafuri, posibil utile și în contextul proiectului nostru, am ales să ne bazăm pe astfel de formate existente. Ele sunt suficient de flexibile pentru scopurile noastre; desigur, aceasta va implica implementarea unor funcții de conversie a datelor de la bănci, operație oricum necesară din cauza variabilității formatelor folosite de bănci.

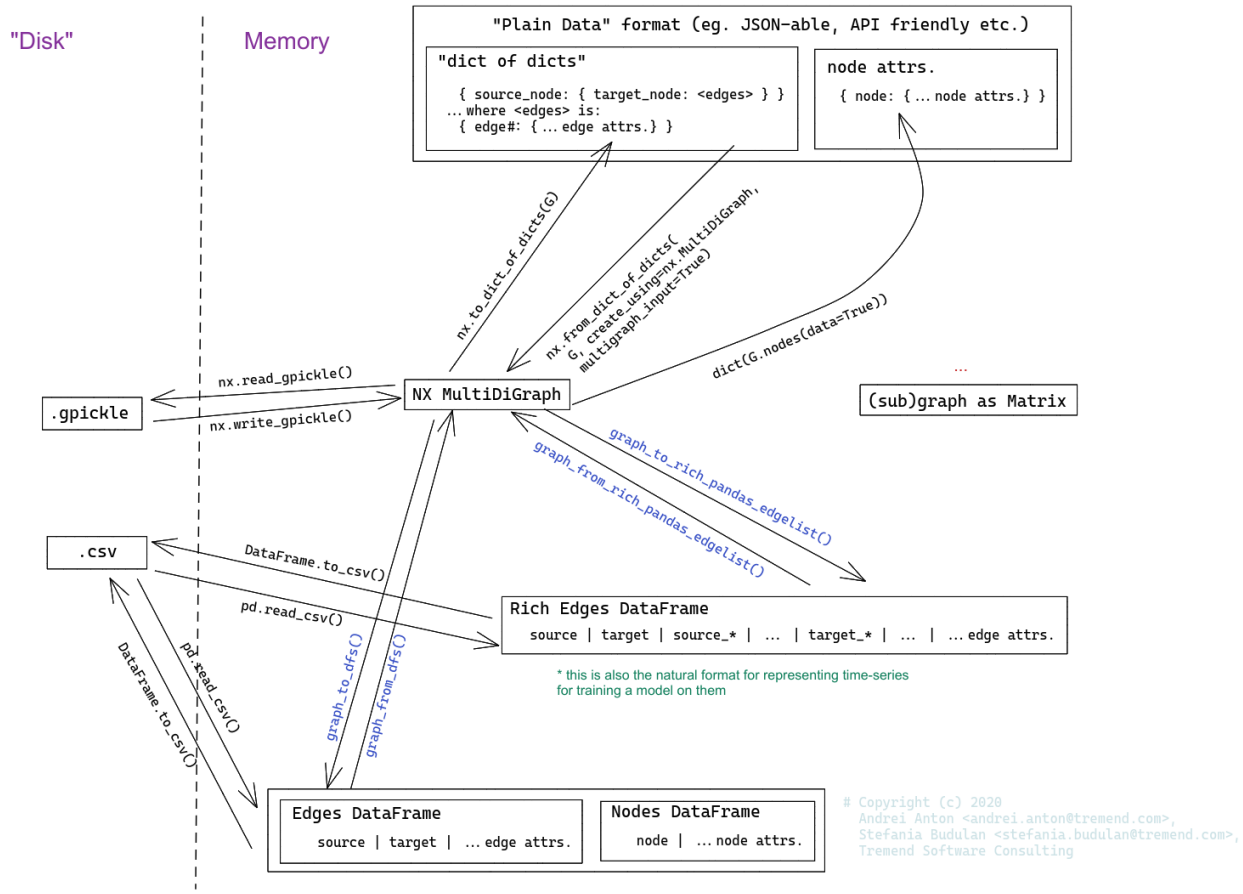

Fig. 1. Formate de date posibile în cadrul bibliotecii

Formatele selectate sunt de două feluri:

- Pentru reprezentarea în memorie
    - Formatul NetworkX (https://networkx.org/) pentru reprezentarea ca graf.
    - Formatul DataFrames (https://pandas.pydata.org/) pentru manipularea grafurilor în reprezentare tabelară; fiecare linie conține nodurile sursă și destinație și atribute pentru arcul care le unește; atributele nodurilor sunt memorate într-un tabel separat. O alta forma de stocare a informatiei de tip graf sub formatul de DataFrames este cea de “Rich Edges”, care conține și atributele nodurilor în același tabel, dar acesta forma este mai putin economică.
    - Formatul "plain data" (transferabil în liste JSON / YAML / XML).
- Pentru reprezentare pe memorie externă
    - `.gpickle` pentru grafuri NetworkX
    - `.csv` pentru DataFrames
    - Posibil și `JSON` pentru "plain data", dacă vom avea nevoie.

Pentru uniformitate și simplitate, la preprocesarea datelor vom pune nodurilor etichete întregi, în secvențe începând de la zero. Formatul DataFrames va fi preferat atunci când este posibil. Listele de atribute vor fi deschise, dar numele câmpurilor uzuale vor fi unice. Pentru atributele de tip categorial rămâne în continuare deschisă problema metodelor de transformare în valori numerice.

1.2 Structura bibliotecii

Descriem aici categoriile de funcții care vor face parte din bibliotecă. Funcțiile de citire/scriere, conversie sau generare de date sunt menționate în secțiunile 1.1 și 2.2. De asemenea, nu vom discuta funcții auxiliare (cum sunt cele de reprezentare a grafurilor), ci ne vom referi doar la funcțiile care implementează operații importante în bibliotecă. Scopul în această etapă a fost inventarierea operațiilor dorite, gruparea lor în categorii pentru care formatul I/O este identic sau similar și stabilirea, într-o primă iterație, a acestui format.

1.2.1 Preprocesarea datelor

Modulul de preprocesare a datelor va conține funcții de prelucrare cu roluri diverse, între care:

- *Verificări de corectitudine a datelor*. Asigurarea unor formate consistente în privința tipurilor datelor. Raportarea eventualelor incongruențe și implementarea unor soluții simple de rezolvare.
- *Tratarea datelor lipsă*. Valorile lipsă pot fi înlocuite cu valori fixe, medii, alte valori statistic relevante, sau pot fi eliminate complet.
- *Standardizarea și conversia valorilor pentru timp și dată*. Momentul tranzacției este un atribut important pentru analiză. Formatele diverse sunt o posibilă sursă de interpretări greșite. Vom folosi un singur format, la care conversia se va face automat.
- *Transformarea câmpurilor categoriale.* Variabilele cu valori discrete, fără semnificație numerică, trebuie convertite într-un vector numeric. Există diverse metode în acest scop (label-encoding, one-hot encoding, term frequency–inverse document frequency). Expansiunea dimensională a spațiului trăsăturilor poate fi ulterior limitată prin aplicarea analizei componentelor principale (PCA).

- *Normalizare și standardizare*.

1.2.2 Extragerea trăsăturilor

Metodele pe care le avem în vedere pentru detecția de anomalii folosesc fie direct structura de graf, fie trăsături extrase din aceasta și din atribute, fie o combinație a celor două abordări. În primul caz, extragerea subgrafurilor sau a altor structuri din graf se va face cu funcții specifice iar analiza va ține seama în mod esențial și direct de topologia grafului.

Extragerea trăsăturilor este o operație tipică în învățarea automată. Scopul ei este reducerea dimensiunii vectorilor (tranzacțiilor, înregistrărilor, etc.) supuși analizei, prin eliminarea componentelor nediscriminative, comprimarea altor componente și, mai ales, înlocuirea informațiilor brute cu altele obținute prin transformări adecvate care scot în evidență caracteristici semnificative ale semnalelor. Vectorii de trăsături sunt apoi transmiși funcțiilor de învățare (cu scopul detecției de anomalii, în cazul nostru).

De exemplu, pentru o structură de graf, în care un vector este un subgraf (de exemplu, corespunzând unei ferestre de timp limitate sau unui număr fix și mic de tranzacții), unele trăsături pot fi [ATK15, Ell19]:

- La nivel de nod: grad (inclusiv pentru multigrafuri), apropiere.
- La nivel local: numărul de arce, densitatea lor, numărul unor structuri fundamentale simple, de exemplu triunghiuri [AMF10].
- La nivel de grup: densitate, modularitate, diverse trăsături rezultate din analiza spectrală.

La acestea se adaugă trăsături extrase din atribute ale tranzacțiilor care nu contribuie la topologia grafului, dar care pot contribui la creșterea eficienței detecției. Aici se pot folosi instrumente tipice de reducere dimensională precum PCA.

În cadrul bibliotecii noastre, vom realiza funcții de extragere a trăsăturilor organizate după cum urmează:

- Funcții elementare pentru fiecare tip de trăsătură în parte. Argumentele de intrare vor fi datele despre tranzacții, sub formă tabelară sau de graf (vezi secțiunea 1.1), iar rezultatul va consta în vectorii de trăsături asociați tranzacțiilor.
- Funcții de compunere a unui vector global de trăsături, pe baza unei liste de trăsături elementare descrisă printr-o structură de variabile logice. Vectorul de trăsături este astfel compus fără ca utilizatorul să trebuiască să se informeze asupra funcțiilor elementare. Vor fi disponibile și unele ‘rețete’ de compunere a trăsăturilor pe baza experienței noastre sau a altor cercetători, pentru utilizatorii care au puține cunoștințe în domeniu.

1.2.3 Detecție de anomalii

Pentru implementarea algoritmilor de detecție de anomalii vom folosi modelul I/O din Scikit Learn [SciLe11], preluat și de recenta bibliotecă Python [PyOD] specializată în detecția de anomalii. PyOD conține majoritatea algoritmilor clasici, dar și abordări mai noi, și va fi folosită pentru comparații în testarea algoritmilor pe care îi vom propune.

Fiecare algoritm de detecție de anomalii este asociat unei clase. Operația de detecție pe setul de antrenare se face cu metoda **fit**. Posibilii parametrii ai algoritmului sunt și ei asociați clasei și au valori implicite prestabilite. Singurul argument obligatoriu de intrare este matricea vectorilor de antrenare (aranjați pe coloane). Desigur, vectorii de antrenare pot fi formați din trăsături extrase din date sau pot avea un caracter mai particular legat de topologia grafului (de exemplu, matrice de incidență organizată pe coloane sau descrisă prin enumerare de arce).

Metoda **fit** calculează și indici de performanță uzuali (vezi secțiunea 1.2.4).

Pentru date din afara setului de antrenare, detectorul antrenat se apelează cu metoda **predict** și furnizează etichete normal/anomalie.

Vom folosi PyOD și ca sursă de inspirație pentru combinații de modele, având în vedere caracterul posibil instabil al algoritmilor de detecție de anomalii. O combinație de algoritmi poate da rezultate mai robuste decât un singur algoritm.

Algoritmii de detecție de anomalii pe care îi avem în vedere sunt în special cei nesupervizați. Din punctul de vedere al argumentelor, funcțiile pentru algoritmi supervizați au în plus la intrare etichetele (binare) ale vectorilor de antrenare.

1.2.4 Testare și evaluarea performanței

Pentru testare vom pregăti probleme relevante în stilul obișnuit al bibliotecilor din domeniul public. Seturile de date vor fi atât artificiale, în special pentru prima fază a testării (cu precădere internă), cât și provenite din date reale (prelucrate pentru eliminarea informațiilor sensibile și pentru anonimizarea completă a persoanelor/entităților implicate în tranzacții). Parte din aceste seturi de date vor fi făcute publice (cu acordul furnizorului primar al datelor), pentru ilustrarea funcționării bibliotecii noastre.

Vor fi create exemple de utilizare după formatul obișnuit: descrierea datelor, separarea în date de antrenare și date de testare, obținerea trăsăturilor (dacă este cazul), stabilirea parametrilor, antrenare, raportare rezultate antrenare, raportare rezultate testare.

Evaluarea performanței se va face cu criteriile obișnuite (acuratețe, precizie, recall, scor F1, curba ROC, etc.) pentru calculul cărora există deja funcții în biblioteci publice, în particular PyOD. Vom adapta eventual interfața, în măsura în care e nevoie pentru datele de tip graf, și actualiza aspectul grafic.

2. Programe de generare și conversie de date

2.1 Generare de date artificiale

Generarea de date artificiale este importantă din două motive: pentru testarea rapidă a algoritmilor proiectați și pentru a suplini caracterul parțial al datelor reale. Chiar dacă băncile cu

care cooperăm ar anonimiza doar strictul necesar din informațiile despre clienți și tranzacții, lipsa datelor de la alte bănci implicate în tranzacții face ca imaginea de ansamblu să fie incompletă.

Tranzacțiile frauduloase sunt simulate prin inserare într-un graf sintetic de dimensiuni mari, atât la nivel de subgraf, cât și la nivel de atribute. Multigraful orientat rezultat simulează o listă de tranzacții desfășurate, de obicei, într-o fereastră de timp precizată.

Graful de bază simulând o rețea de tranzacții normale este generat folosind modelul bloc stochastic [HLL83]. Modelul constă în *K* module (blocuri) interconectate conform unei matrice de probabilități ***P***. Alegerea este justificată de asemănarea dintre module și comunități reale. Matricea de probabilități determină măsura în care două noduri pot fi asociate:

- diagonala lui ***P*** determină probabilitatea ca două noduri din același modul să fie conectate
- elementele nediagonale ale lui ***P*** determină probabilitatea ca două noduri din module diferite să fie conectate.

Graful este caracterizat de diverși parametri, precum numărul de noduri, numărul de module, dimensiunile minime și maxime ale modulelor, probabilități minime și maxime ca două noduri să fie conectate, etc.

Anomaliile structurale de mărimi variabile sunt inserate în poziții aleatoare din graf. Am implementat următoarele anomalii relevante pentru fraude bancare [Ell19]:

a) **Inel și cvasi-inel**; sume, de obicei mari, sunt trecute prin mai multe noduri doar pentru a se întoarce la nodul inițial.
b) **Clică**, în care suma poate fi spartă în cantități mai mici care trec prin mai multe noduri interconectate.
c) **Stea**, în care un nod central fie colectează fie distribuie o sumă de la, respectiv către mai multe noduri

Figure 2 ilustrează aceste structuri. La inserarea unei anomalii, mai multe noduri ale grafului de bază sunt selectate aleator; lor le sunt adăugate arce conform anomaliei alese; arcele existente sunt păstrate. Există parametri descriind numărul de anomalii din fiecare tip, dimensiunile lor, posibilitatea suprapunerii.

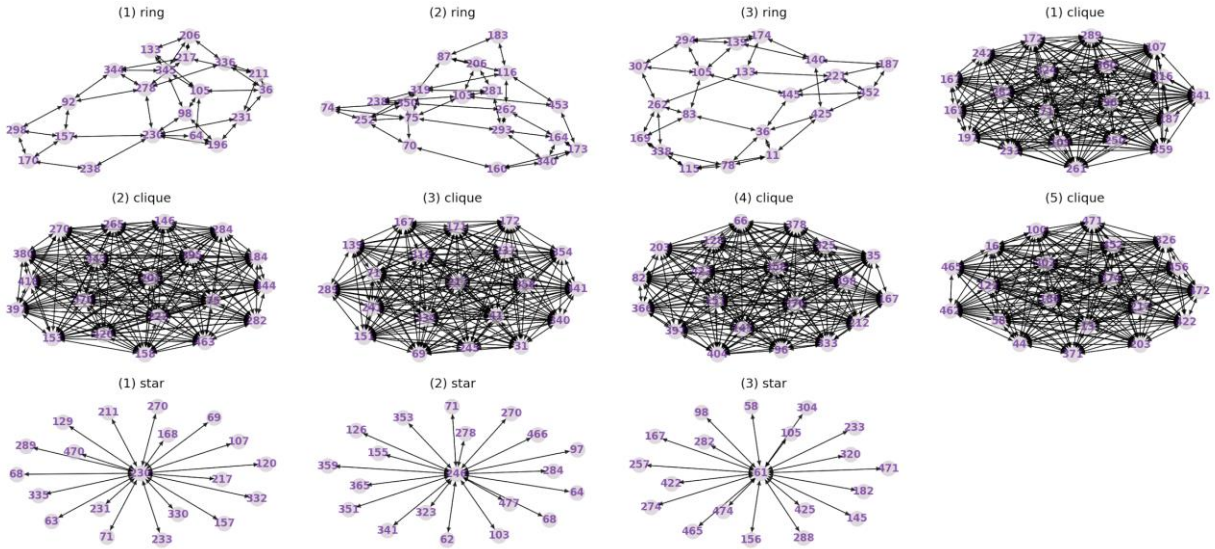

Fig 2: Grafuri cu anomalii de tip inel (subfig.1-3), clică (4-8) și stea (9-12)

Atributele nodurilor și arcelor sunt generate arbitrar, după mai multe distribuții de probabilitate posibile. Suma și momentul tranzacției pot fi controlate mai precis, în concordanță cu tipul de anomalie. Sunt permise variații de la anomalia “teoretică”: de exemplu, suma circulată între noduri nu este identică la toate tranzacțiile, ci poate suferi diverse variații.

Toate funcțiile de generare descrise mai sus au fost implementate și testate.

2.2 Conversie de date

După cum este precizat în Secțiunea 1.1, am ales mai multe formate de reprezentare a datelor de tip graf, pentru a profita de bibliotecile existente. Fig.1 arată aceste tipuri de date (în căsuțe dreptunghiulare) și funcțiile de conversie între ele (cu săgeți). Funcțiile scrise cu albastru sunt realizate de noi, cele cu negru sunt existente în biblioteci. Funcțiile sunt implementate într-un modul numit data_conversion; numele funcțiilor arată direct funcționalitatea. Avem deci libertatea de a ne mișca între formate după cum se va dovedi util pentru detecția de anomalii.

De asemenea, am realizat machete pentru funcțiile de conversie ale datelor venind din domeniul bancar, cu exemple preliminare pentru primele astfel de seturi de date (BRD, Libra).

3. Programe de detecție de comunități

Noțiunea de comunitate într-un graf are un sens larg, cel mai comun înțeles fiind cel în care nodurile din comunitate sunt bine conectate în timp ce conexiunile cu alte noduri sunt mai rare. O noțiune înrudită este cea de clusterizare (în grafuri) [FoSa10]. Comunitățile sunt disjuncte sau pot exista suprapuneri.

La nivel general, detecția de comunități (DC) constă în [YaLe15]:

- alegerea unei **funcții scor** prin care se calculează un scor asociat unei mulțimi de noduri sau unei partiții a unui graf
- un algoritm eficient pentru determinarea partiției grafului în comunități, astfel încât scorul global este maximizat.

În contextul proiectului nostru, DC poate servi la două scopuri:

- **Detecția directă a anomaliilor în comunități** - după obținerea comunităților este ușor de decis care sunt normale și care sunt anomalii. Algoritmii folosiți nu au un număr impus de comunități.
- **Partiționarea în subgrafuri a grafurilor prea mari pentru prelucrare** - în acest caz detecția de comunități are rol de preprocesare și algoritmii folosiți au mai degrabă un număr impus de comunități. Anomaliile se detectează ulterior în interiorul comunităților.

Scopul nostru în această etapă a fost explorarea algoritmilor existenți, alegerea celor mai promițători dintre ei și realizarea unei interfețe unice prin care să putem apela algoritmi din diverse surse.

3.1 Programe și biblioteci existente

Pentru a folosi programe care au fost validate de utilizatori, am selectat următoarele biblioteci:

1. modulul communities din biblioteca **NetworkX** [HSC08], care conține implementări ale unor algoritmi DC fără suprapuneri: Girvan-Newman [GiNe02], Clauset-Newman-Moore [CNM04], Label Propagation (două variante) [RAK07], Fluid Communities [Pa17], Kernighan–Lin Bipartition [KeLi70];
2. biblioteca python-louvain/community conținând cunoscuta metodă Louvain [BGL08];
3. biblioteca Karate Club [RKS20] conține zece algoritmi DC, care însă nu pare a fi ajuns la o formă stabilă, cu documentație parțial inconsistentă și comportament uneori neașteptat;
4. biblioteca CDLib [GTB18] conține mulți algoritmi DC, însă neunificați printr-o interfață comună și aparent cu valori implicite ale parametrilor care nu sunt suficient de generale;
5. biblioteca leidenalg implementează algoritmul Leiden CD.

3.2 Evaluarea algoritmilor de detecție de comunități

Se pot utiliza mai multe metode pentru a evalua performanțele algoritmilor DC:

- Prin comparație cu grafuri cu comunități reale etichetate. Astfel de grafuri pot fi obținute prin generare artificială (vezi secțiunea 2.1) sau sunt disponibile pentru anumite modele: rețele sociale, rețele de articole sau reviste și citări, grafuri de pagini web. Adecvarea rezultatului CD la realitate se măsoară cu Rand Index (RI), Adjusted Rand Index (ARI) sau Mutual Information (MI) Score.
- Direct, prin măsuri caracterizând comunitățile obținute, de exemplu separabilitate și densitate [YaLe15]. Desigur, aceste măsuri sunt oarecum artificiale și caracterizează bine mai ales grafurile generate.
- Indirect, prin măsurarea efectului unei operații ulterioare, precum detecția de anomalii.

În contextul celei de-a treia metode, propunem o măsură a păstrării subgrafurilor de anomalii în aceeași comunitate: calculul Rand Index doar pe subgraful indus de nodurile anormale. Mai

precis, este vorba despre fracția de arce între noduri anormale care sunt separate în comunități diferite (ACS-ARI = Anomalies vs. Communities Subgraph Adjusted Rand Index).

Pentru testare am folosit trei grafuri “adevărate”:

- Artificiale, cu 2.5k noduri și 5 comunități.
- ogbn-arxiv https://ogb.stanford.edu/docs/nodeprop/#ogbn-arxiv , ~170K noduri, 1.1M arce, 40 communities, în care nodurile sunt articole, arcele citări, iar comunitățile sunt domeniile științifice.
- Rețeaua socială Youtube, https://snap.stanford.edu/data/com-Youtube.html, ~1M noduri, 3M arce, 8K comunități, în care nodurile sunt utilizatori, arcele relații de prietenie, iar comunitățile sunt grupuri create de utilizatori.

3.3 Rezultate

Dăm aici doar concluziile testelor, fără rezultate numerice (cuprinse într-un document intern).

Algoritmul Louvain [BGL08] are comportament foarte bun pe toate seturile de date testate. Este robust și bine testat și de alți utilizatori, de aceea poate fi prima noastră opțiune în viitor.

Algoritmul EdMot [LHWL19] din KarateClub dă cele mai bune rezultate pe grafuri artificiale și conservă bine anomaliile.

Algoritmul Informap (CDLib), bazat pe o abordare din teoria informației, are rezultate bune, dar ușor inferioare algoritmilor Louvain și EdMot.

Algoritmul Label Propagation, în varianta [CoGa10] implementată în NetworkX, este cel mai rapid și dă rezultate mulțumitoare în privința ARI; totuși, nu pare a fi robust, rezultatele au variabilitate mare.

Algoritmul Girwan-Newman (NetworkX), deși binecunoscut, este cel mai lent și dă rezultate mai slabe decât Louvain.

Alți algoritmi testați au fost SymmNMF (Karate Club), Demon (CDLib), BigClan (Karate Club), EgoNetSplitter (Karate Club), dar rezultatele slabe nu îi recomandă pentru utilizare ulterioară.

3.4 Interfață unică pentru integrarea în biblioteca noastră

Bibliotecile pentru detecție de comunități prezentate mai sus au interfețe diferite, deși rezolvă aceeași problemă. Am dezvoltat o interfață unică bazată pe o organizare în stilul celei folosite și pentru alte probleme din proiect: fiecare algoritm are asociată o clasă. Modul de apel de bază este identic pentru fiecare algoritm. Parametrii fiecărei metode pot fi accesați similar, dar desigur că au semnificații diferite. În acest fel, dezvoltările ulterioare pot beneficia de caracterul modular al implementării și pot astfel folosi cu ușurință algoritmul CD cel mai potrivit în contextul respectiv.

4. Versiune Python pentru biblioteca DL

Învățarea dicționarelor (Dictionary learning - DL) [DI18] este un instrument care profită de versatilitatea reprezentărilor rare pentru a rezolva multe probleme de prelucrarea semnalelor și învățare automată, începând de la eliminarea zgomotului și mergând până la clasificare și detecție de anomalii. Dat fiind un set de *N* semnale de antrenare de dimensiune *m,* puse pe coloanele unei matrice ***Y***, problema DL constă în găsirea unui dicționar ***D*** de dimensiune *mxn* și a unei matrice rare ***X*** de dimensiune *nxN*, având cel mult *s* elemente pe fiecare coloană, astfel încât produsul ***DX*** este o bună reprezentare a lui ***Y***. Odată ce un astfel de dicționar este obținut, un semnal ***y*** aparținând aceleiași familii ca semnalele de antrenare poate fi reprezentat sub forma ***Dx***, unde ***x*** este un vector rar cu ***s*** elemente care poate fi găsit cu Orthogonal Matching Pursuit (OMP) sau alți algoritmi de reprezentare rară. Vectorul ***Dx*** poate fi interpretat ca fiind semnalul "adevărat" în contextul eliminării zgomotului, presupunând că DL produce un dicționar care surprinde esența familiei de semnale studiate. În clasificare, vectorul rar ***x***, numit și reprezentare, poate fi folosit pentru a decide clasa de care aparține semnalul.

Există mulți algoritmi pentru DL. Cartea [DI18] este însoțită de o bibliotecă de programe MATLAB (https://github.com/pirofti/dl-box) implementând metode semnificative, exemple și aplicații. Unul din obiectivele proiectului Graphomaly este de a implementa o versiune completă în Python a acestei biblioteci. Acest obiectiv a fost îndeplinit. Au fost adăugate și unele funcții noi de clasificare și pentru lucrul cu nuclee. Testarea este aproape completă și va fi încheiată până la sfârșitul anului. Versiunea curentă este la dispoziția tuturor membrilor echipei. Biblioteca va fi în continuare îmbogățită pe măsură ce se vor identifica algoritmi de interes, dar deja poate fi utilizată pentru dezvoltarea unor metode de detecție de anomalii.

Biblioteca nu a primit încă un nume, de aceea o vom numi generic **Python DL Toolbox**. Ea conține algoritmi pentru trei scenarii diferite: **standard DL**, **online DL** și **kernel DL**. Vom prezenta în continuare principalele metode implementate.

4.1 Algoritmi DL implementați

Standard DL este problema de bază descrisă mai sus, în care țelul este cea mai bună aproximare a semnalelor de antrenare ***Y*** prin reprezentarea rară dată de produsul ***DX***. Algoritmii au caracter iterativ, în fiecare iterație actualizându-se succesiv reprezentările ***X***, cu OMP, și dicționarul ***D***. Am implementat următoarele metode de actualizare a dicționarului: **k-svd**, **ak-svd**, **naksvd**, **sgk**, **nsgk** și **mod**. Există, de asemenea, versiuni potențial paralele, în care mai mulți atomi ai dicționarului sunt actualizați simultan. În plus, fiecare algoritm este adaptat și pentru versiunile problemei DL cu **regularizare** și **coerență**. Regularizarea induce folosirea unor coeficienți mai mici în matricea de reprezentare ***X***, ceea ce îmbunătățește indirect condiționarea dicționarului. Cealaltă modificare a problemei DL are în vedere creșterea incoerenței între atomii dicționarului (atomii sunt mai aproape de ortogonalitate și mai departe de colinearitate). În ambele cazuri, deși eroarea de reprezentare este mai mare pe setul de antrenare, robustețea dicționarului obținut permite reprezentări mai bune pe semnale noi.

Online DL este varianta problemei standard în care semnalele **Y** nu sunt disponibile de la început sau numărul lor este foarte mare și deci sunt imposibil de procesat de algoritmii standard din cauza cerințelor de memorie sau de calcul. Dicționarul este actualizat fie la fiecare semnal primit, fie în batch-uri (conținând mai mult de un semnal, dar mai puțin de *N*). În acest sens, algoritmii pentru problema standard pot fi văzuți ca online DL cu batch de dimensiune *N*. Am implementat două metode, anume **Online Coordinate Descent DL** și **Recursive Least Squares DL**.

Metodele **kernel DL** au scopul de a trece dincolo de caracterul liniar al reprezentării. Spațiul semnalelor este transformat neliniar într-un spațiu al trăsăturilor. Fiecărui semnal $\boldsymbol{y_i}$ i se asociază un vector $\boldsymbol{\varphi(y)}$ de trăsături, unde $\boldsymbol{\varphi(y)}$ este o funcție neliniară. Dicționarul $\boldsymbol{D}$ este extins la spațiul neliniar și capătă forma $\boldsymbol{\varphi(Y)A}$, unde $\boldsymbol{A}$ conține acum elementele dicționarului. Problema de optimizare este transformată prin utilizarea nucleelor Mercer, ceea ce permite înlocuirea produsului scalar a doi vectori de trăsături prin calculul unei funcții nucleu $\boldsymbol{k}$. Problema de optimizare rezultată este rezolvată prin transformări ale algoritmilor standard. Am implementat versiuni kernel pentru toți algoritmii standard menționați mai sus. Deoarece utilizarea nucleelor crește dimensiunea problemei de optimizare, din cauza utilizării matricei $\boldsymbol{K=k(Y,Y)=\varphi(Y)^T\ \varphi(Y)}$, care are dimensiune *N*x*N*, sunt utile metode de aproximare de rang mic a matricei nucleu K, prin care se obțin soluții mai generale decât cele liniare, dar cu complexitate mai mică decât kernel DL. Din această categorie am implementat metodele bazate pe eșantionare Nystrom [GoEl16] și trăsături aleatoare [RaBe07].

Funcțiile care implementează cele trei metode DL discutate mai sus au sintaxa:

- dictionary_learning(Y, D0, n_nonzero_coefs, n_iterations,
  coding_method, learning_method, params)
- online_dictionary_learning(Y, D0, n_nonzero_coefs, n_iterations,
  coding_method, learning_method, params)
- kernel_dictionary_learning(Y, A0, n_nonzero_coefs, n_iterations,
  coding_method, learning_method, kernel_method, params).

Variabilele de intrare din aceste funcții sunt:

- Y - matricea semnalelor de antrenare
- D0, A0 - dicționarele inițiale (pot fi alese aleator prin mai multe metode)
- n_nonzero_coefs - numărul de coeficienți nenuli din reprezentare, notat cu *s* mai sus
- coding_method - metoda prin care se calculează reprezentările rare (în primul rând OMP - Orthogonal Matching Pursuit, dar se pot utiliza și alte metode)
- learning_method - metoda de actualizare a dicționarului în procesul de optimizare (k-svd, ak-svd, etc.)
- kernel_method - funcția nucleu cu care se calculează matricea nucleu $\boldsymbol{K}$
- params - structură conținând parametrii problemei de optimizare și ai procesului de optimizare.

Funcțiile întorc dicționarele optimizate, reprezentările asociate și diverse măsuri ale calității rezultatului (de exemplu, erori). Avem în vedere realizarea unei interfețe unice pentru toate funcțiile de antrenare. Există și alte funcții auxiliare care pot fi folosite pentru evaluare.

Structura bibliotecii permite asamblarea de către utilizator a algoritmului de optimizare și stabilirea parametrilor importanți prin variabila **params**. Desigur, există valori implicite pentru toți parametrii, biblioteca putând fi astfel folosită de utilizatori cu grade diferite de experiență.

4.2 Algoritmi de clasificare

În plus față de biblioteca MATLAB, am implementat mai mulți algoritmi de clasificare ce folosesc DL. Este vorba despre: Dictionary Pair Learning (DPL) [GuSu14], Discriminative Dictionary Learning (DDL) [PhVe08], [ZhLi10] și Label Consistent Dictionary Learning (LCDL) [JiLi13]. Cei trei algoritmi au o interfață similară modelului Scikit Learn [SciLe11]. Fiecare algoritm este asociat unei clase (în sensul din limbajele de programare). O problemă de clasificare este rezolvată în doi pași: învățare (cu metoda **fit**) și testare (cu metoda **predict**).

Pentru toți cei trei algoritmi există posibilitatea de a trata problemele în stil online, în sensul că datele sunt furnizate în batch-uri. Acest mod de lucru pare a fi benefic pentru rezultatele clasificării.

Cei trei algoritmi au fost extinși și la metode de tip kernel. Există deci trei metode de clasificare de acest tip: KDPL, KDDL și KLCDL. Dacă cele două din urmă apelează direct metode DL, pentru prima a fost necesară o implementare separată.

4.3. Alte funcții ale bibliotecii

Python DL Toolbox conține mai multe funcții auxiliare, între care menționăm pachetul **datasets**, prin care pot fi încărcate seturi de date standard. Pachetul a fost inspirat de biblioteca Python Tensorflow / Keras. Seturile de date sunt disponibile prin Amazon Web Service și pot fi descărcate prin următoarele două linii de cod (de exemplu aici pentru Yale B):

```
from datasets import yaleb
(X_train, y_train), (X_test, y_test) = yaleb.load_data()
```

Deocamdată seturile de date sunt cele folosite pentru probleme de clasificare. Avem în plan extinderea la seturi de date provenind din probleme de detecție de anomalii, în special în grafuri.

5. Alte informații despre proiect

Am realizat site-ul proiectului, http://graphomaly.upb.ro. Deocamdată conține informații minimale despre proiect și link-uri către parteneri, dar va fi dezvoltat pe măsura obținerii rezultatelor.
Echipa a cooperat bine până acum, sarcinile de lucru fiind distribuite în bună înțelegere și rezolvate cu multă inițiativă. Avem ședințe tehnice săptămânale, ceea ce a contribuit nu doar la progres în atingerea obiectivelor dar și la cunoașterea reciprocă. Colaborare software are loc pe gitlab; deocamdată nu am făcut public niciun program; primul candidat este biblioteca DL. Cooperarea cu partenerii externi pentru obținerea de date s-a desfășurat puțin mai lent decât

estimam inițial, dar avem un set de date de la BRD și am deschis o colaborare cu Libra, care promite lucrul pe seturi de date extinse și adnotate.

6. Concluzii

Apreciem că în cele patru luni de la începerea oficială a proiectului am realizat ce ne-am propus pentru această etapă. Am construit bazele pe care să dezvoltăm algoritmii de detecție de anomalii, adică să atacăm obiectivele prevăzute pentru anul viitor. Avem deja premize pentru realizarea unor publicații, dar este prematur să discutăm posibilul conținut al acestora.

Bibliografie

[AMF10] L. Akoglu, M. McGlohon, C. Faloutsos.“Oddball: Spotting anomalies in weighted graphs”. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, 410–421, 2010.

[ATK15] L. Akoglu, H. Tong, D. Koutra. "Graph based anomaly detection and description: a survey." *Data mining and knowledge discovery* 29, no. 3, 626-688, 2015.

[BGL08] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre. Fast unfolding of communities in large networks, J. Stat. Mech., 10, P10008, Oct. 2008.

[CoGa10] G. Cordasco and L. Gargano, “Community detection via semi-synchronous label propagation algorithms,” in IEEE International Workshop on: Business Applications of Social Network Analysis (BASNA), 1-8, Dec. 2010.

[CNM04] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks,” Phys. Rev. E, 70(6), p. 066111, Dec. 2004.

[DI18] B.Dumitrescu, P.Irofti - Dictionary Learning Algorithms and Applications, Springer, 2018.

[Ell19] A. Elliott et al. Anomaly detection in networks with application to financial transaction networks. *arXiv:1901.00402*, 2019.

[FoSa10] S. Fortunato, Community detection in graphs, Physics Reports, vol. 486, no. 3–5, pp. 75–174, Feb. 2010.

[GiNe02] M. Girvan and M. E. J. Newman. Community structure in social and biological networks, PNAS, 99(12), 7821–7826, Jun. 2002.

[GoEl16] A. Golts, M. Elad, Linearized kernel dictionary learning. IEEE J. Sel. Top. Signal Process. 10(4), 726–739, 2016.

[GTB18] F. Giannotti, R. Trasarti, K. Bontcheva, V. Grossi. SoBigData: Social Mining & Big Data Ecosystem, Proc. of The Web Conference, Geneva, 437–438, Apr. 2018.

[GuSu14] S. Gu, et al. "Projective dictionary pair learning for pattern classification." Advances in neural information processing systems, 793-801, 2014.

[HLL83] P. W. Holland, K. B. Laskey, and S. Leinhardt, “Stochastic blockmodels: First steps,” Social Networks, vol. 5, no. 2, pp. 109–137, Jun. 1983.

[HSC08] A. Hagberg, P. Swart, D. S Chult. Exploring network structure, dynamics, and function using networkx, Los Alamos National Lab. (LANL), LA-UR-08-05495, Jan. 2008. Accessed: Nov. 23, 2020. [Online]. Available: https://www.osti.gov/biblio/960616.

[JiLi13] Z. Jiang, Z. Lin, L.S. Davis, Label consistent K-SVD: learning a discriminative dictionary for recognition. IEEE Trans. Pattern Anal. Mach. Intell. 35(11), 2651–2664, 2013.

[KeLi70] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs, The Bell System Technical Journal, 49(2), 291–307, Feb. 1970.

[LHWL19] P.-Z. Li, L. Huang, C.-D. Wang, J.-H. Lai. EdMot: An Edge Enhancement Approach for Motif-aware Community Detection,” Proc. 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 479–487, 2019.

[Pa17] F. Parés et al., “Fluid Communities: A Competitive, Scalable and Diverse Community Detection Algorithm,” arXiv:1703.09307 [physics], Oct. 2017, Accessed: Nov. 23, 2020. [Online]. Available: http://arxiv.org/abs/1703.09307.

[PhVe08] D.S. Pham, S. Venkatesh, Joint learning and dictionary construction for pattern recognition, in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[PyOD] Y.Zhao, Z. Nasrullah, Z. Li. PyOD: A Python Toolbox for Scalable Outlier Detection. Journal of Machine Learning Research, 20(96), 1-7, 2019.

[RAK07] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E, 76(3), 036106, Sep. 2007.

[RaBe07] A.Rahimi, B. Recht. "Random features for large-scale kernel machines." Advances in neural information processing systems 20, 1177-1184, 2007.

[RKS20] B. Rozemberczki, O. Kiss, and R. Sarkar, “Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs,” arXiv:2003.04819 [cs, stat], Aug. 2020, Accessed: Oct. 14, 2020. [Online]. Available: http://arxiv.org/abs/2003.04819.

[SciLe11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas. Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research, 12, 2825-2830, 2011.

[YaLe15] J. Yang, J. Leskovec. Defining and evaluating network communities based on ground-truth. Knowledge and Information Systems, 42(1), pp.181-213, 2015.

[ZhLi10] Q. Zhang, B. Li, Discriminative K-SVD for dictionary learning in face recognition, in Proceedings of IEEE Conf. Computer Vision and Pattern Recognition, 2691–2698, 2010.

Director proiect,
Prof. Bogdan Dumitrescu