VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

APLIKACE METODY UČENÍ BEZ UČITELE NA HLEDÁNÍ PODOBNÝCH GRAFŮ

APPLICATION OF UNSUPERVISED LEARNING METHODS IN GRAPH SIMILARITY SEARCH

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **JOZEF SABO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. ZBYNĚK KŘIVKA, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání diplomové práce

Student: **Sabo Jozef, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Vývoj aplikací
Název: **Aplikace metody učení bez učitele na hledání podobných grafů**
Application of Unsupervised Learning Methods in Graph Similarity Search
Kategorie: Bezpečnost
Zadání:

1. Seznamte se s metodami učení bez učitele a jejich možné aplikaci na grafové struktury. Seznamte se s grafovou strukturou pro popis chování systému, kterou používá firma Avast.
2. Studujte již existující metody, které by mohly být nápomocny při řešení tématu.
3. Navrhněte architekturu samoučícího se systému pro analyzování grafů chování a hledání podobností v těchto grafech.
4. Navržený systém implementujte.
5. Implementovaný systém experimentálně vyhodnoťte a diskutujte další možný vývoj.

Literatura:

- Anton Tsitsulin, et al: Graph Clustering with Graph Neural Networks. In: MLG'20, August 24, 2020, ACM. Dostupné on-line na http://www.mlgworkshop.org/2020/papers/MLG2020_paper_42.pdf [cit. 2020-10-14]
- Bai, Yunsheng & Ding, Hao & Bian, Song & Sun, Yizhou & Wang, Wei. (2018). Graph Edit Distance Computation via Graph Neural Networks. Dostupné on-line na https://www.researchgate.net/publication/327110628_Graph_Edit_Distance_Computation_via_Graph_Neural_Networks [cit. 2020-10-12]
- Karate club - Unsupervised machine learning library for NetworkX. Dostupné on-line na https://karateclub.readthedocs.io/en/latest/index.html [cit. 2020-10-12]

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2, 3 a rozpracování bodu 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Křivka Zbyněk, Ing., Ph.D.**
Konzultant: Drbal Miroslav, Ing., Avast
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 22. října 2020

Abstrakt

Cieľom tejto diplomovej práce bolo v spolupráci s firmou Avast navrhnúť systém, ktorý dokáže dolovať znalosti z databázy grafov pomocou metód učenia bez učiteľa. Grafy, určené pre dolovanie, popisujú chovanie počítačových systémov a do databázy prichádzajú anonymne od používateľov softvérových produktov firmy. Grafom v databáze je možné priradiť jednu z dvoch tried: čistý graf alebo malware (škodlivý) graf. Úlohou navrhnutého samoučiacieho systému je nad grafovou databázou nájsť zhluky grafov, v ktorých sa triedy grafov nemiešajú. Zhluky grafov, v ktorých sa nachádza iba jedna trieda grafov, sa dajú interpretovať ako rôzne typy čistých alebo malware grafov a sú užitočným zdrojom ďalších analýz nad grafmi. Pre ohodnotenie kvality zhlukov bola navrhnutá vlastná metrika pomenovaná ako jednofarebnosť. Metrika hodnotí kvalitu zhlukov na základe toho ako veľmi sa v zhlukoch miešajú čisté a malware grafy. Najlepšie výsledky metrika dosiahla, keď boli vektorové reprezentácie grafov vytvorené modelom hlbokého učenia (variačným grafovým autoenkodérom s dvomi relačnými grafovými konvolučnými operátormi) a pre zhlukovanie nad vektormi bola použitá bezparametrická metóda MeanShift.

Abstract

Goal of this master's thesis was in cooperation with the company Avast to design a system, which can extract knowledge from a database of graphs. Graphs, used for data mining, describe behaviour of computer systems and they are anonymously inserted into the company's database from systems of the company's products users. Each graph in the database can be assigned with one of two labels: clean or malware (malicious) graph. The task of the proposed self-learning system is to find clusters of graphs in the graph database, in which the classes of graphs do not mix. Graph clusters with only one class of graphs can be interpreted as different types of clean or malware graphs and they are a useful source of further analysis on the graphs. To evaluate the quality of the clusters, a custom metric, named as monochromaticity, was designed. The metric evaluates the quality of the clusters based on how much clean and malware graphs are mixed in the clusters. The best results of the metric were obtained when vector representations of graphs were created by a deep learning model (variational graph autoencoder with two relation graph convolution operators) and the parameterless method MeanShift was used for clustering over vectors.

Kľúčové slová

graf, Avast, učenie bez učiteľa, zhlukovanie, detekcia komunít, vektorové reprezentácie uzlov, vektorové reprezentácie grafov, grafové neurónové siete

Keywords

graph, Avast, unsupervised learning, clustering, communities detection, node embeddings, graph embeddings, graph neural networks

Citácia

SABO, Jozef. *Aplikace metody učení bez učitele na hledání podobných grafů*. Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Zbyněk Křivka, Ph.D.

Aplikace metody učení bez učitele na hledání podobných grafů

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením, konzultanta z firmy Avast, Ing. Miroslava Drbala a vedúceho práce z FIT VUT v Brně, Ing. Zbyňka Křivku, Ph.D.. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Jozef Sabo
24. mája 2021

Poďakovanie

Týmto by som sa chcel poďakovať Ing. Miroslavovi Drbalovi a Ing. Zbyňkovi Křivkovi, Ph.D. za ich odborné vedenie a cenné rady pri tvorbe tejto práce.

Obsah

Kapitola 1

Úvod

Žijeme vo svete, ktorého prvky sú často pospájané rozličnými vzťahmi. Príkladom z chémie sú molekuly, ktoré sú zložené z atómov pospájaných chemickými väzbami. Príkladom z nášho oboru sú počítačové siete, kde sú počítače prepojené s inými počítačmi po celom svete rôznymi zariadeniami a prepojeniami. Ak chceme objekty vo vzťahoch skúmať a získavať z nich nové znalosti je potrebné ich nejako v počítačoch reprezentovať. Existujú viaceré možnosti reprezentácie. Napríklad SMILES reťazce dokážu popísať zloženie molekuly [1]. Takýmto popisom molekuly však stratíme mnoho užitočných informácií o jej štruktúre pri tvorbe modelu pre dolovanie znalostí [17]. Najlepšou možnou reprezentáciou, ktorá zachováva všetky vzťahy medzi objektmi v ich originálnej podobe sú grafy. Formálny popis grafov sa nachádza v podkapitole 2.1. Predstavenie všetkých dolovacích úloh, ktoré boli prevedené nad grafmi v tejto práci, sa nachádza v podkapitole 2.2.

Firma Avast využíva grafy pre popis chovania počítačových systémov (ďalej sú v práci takéto grafy pomenované v skratke iba ako behaviorálne grafy) a vedie si databázu takýchto grafov. Databáza je postupne anonymne obohacovaná o nové grafy zo zachytených stavov systémov od používateľov ich softvérových produktov. Behaviorálne grafy sú podrobne popísané v samostatnej podkapitole 5.1. Pre vysvetlenie ďalšieho obsahu práce v tejto úvodnej kapitole je však potrebné minimálne vysvetliť, že každý behaviorálny graf sa skladá z dvoch hlavných typov uzlov: uzlov reprezentujúcich vykonateľné súbory (v OS MS Windows napr. súbory s príponou ‘.exe‘) a uzlov reprezentujúcich procesy OS (inštancie vykonateľných súborov). Uzlom behaviorálnych grafov je možné priradiť jednu z dvoch tried: čistá (neškodná) trieda alebo malware (škodlivá) trieda. Keďže všetky možné experimenty vykonané v práci boli prevedené aj na úrovni uzlov a aj na úrovni celých grafov (prípadne podgrafov), tak bol vymyslený spôsob ako aj grafom taktiež priradiť jednu z týchto dvoch tried. Za malware graf bol označený graf s aspoň jedným malware uzlom a graf bez malware uzlov bol označený za čistý graf.

Hlavnou úlohou tejto diplomovej práce bolo navrhnúť a implementovať systém, ktorý vie dolovať znalosti z databázy behaviorálnych grafov iba pomocou metód učenia bez učiteľa. V práci navrhnutý samoučiaci systém sa nad grafovou databázou snaží nájsť zhluky, v ktorých sa triedy prvkov (pre zjednodušenie zápisov sa prvkami v celej práci myslia uzly alebo grafy) nemiešajú. Zhluk prvkov, v ktorom sa nachádza iba jedna trieda, sa dá interpretovať ako špecifickejší prípad jednej z tried. Napríklad, v prípade zhluku procesových uzlov by jeden z možných zhlukov mohol reprezentovať čisté procesy balíka Microsoft Office. Identicky, v prípade zhluku grafov by jeden z možných zhlukov mohol reprezentovať, napríklad malware behaviorálne grafy, ak sa v systéme nachádza Trojský kôň. Takto najdené zhluky sú užitočným zdrojom ďalších analýz nad grafmi. V zhlukoch je možné napríklad

skúmať aké spoločné vlastnosti majú prkvy zhluku. Pre ohodnotenie kvality zhlukov bola navrhnutá vlastná metrika pomenovaná ako jednofarebnosť. Metrika hodnotí kvalitu zhlukov na základe toho ako veľmi sa v zhlukoch miešajú čisté a malware prvky. Čím menej sa čisté a malware prvky v zhlukoch miešajú, tým lepšie je ohodnotenie metrikou. Je dobré si uvedomiť, že štítky prvkov sú použité iba pre ohodnotenie kvality zhlukov a nie pre samotné trénovanie. Podrobnejšie je táto metrika popísaná v podkapitole 5.3.

Samoučiaci systém bol v práci samostatne navrhnutý pre grafy a aj pre ich najmenšie časti – uzly. Navrhnutý samoučiaci systém hľadajúci "jednofarebné"zhluky nad uzlami sa skladá z dvoch častí. Prvou časťou je model hlbokého učenia, ktorý vytvára vektorové reprezentácie uzlov. Druhou časťou je klasická zhlukovacia metóda, ktorá je schopná nad vektorovými reprezentáciami uzlov nájsť zhluky. Navrhnutý systém pre hľadanie "jednofarebných"zhlukov nad grafmi obsahuje naviac ešte časť, ktorá graf, na vstupe systému, dekomponuje na menšie podgrafy pomocou metódy pre detekciu komunít. Vektorové reprezentácie sú potom vytvárané nad podgrafmi a nie nad celými grafmi. Dôvod za rozdelím grafov na menšie časti je v tom, že grafová reprezentácia počítačového systému napadnutého vírusom, nikdy nebude obsahovať iba malware uzly. Aplikovaním metódy pre detekciu komunít nad grafom, chceme separovať časti grafu obsahujúce malware a časti grafu obsahujúce čisté uzly. Podrobnejší popis a blokové schémy navrhnutých systémov sa nachádzajú v podkapitole 5.2.

Vybratie konkrétnych metód, pre každú z častí navrhnutého systému tak, aby zhluky na výstupe systému dosahovali najlepšej hodnoty metriky jednofarebnosť, bolo predmetom experimentovania, ktoré je opísané v podkapitole 6.1. Potrebná teória, pre pochopenie použitých metód učenia bez učiteľa, sa nachádza v kapitole 4.

Okrem metód učenia bez učiteľa boli v práci vykonané aj doplnkové experimenty s modelmi učenia s učiteľom. Tieto experimenty boli vykonané pre zistenie toho, či je možné vylepšiť kvalitu výsledkov modelov učenia s učiteľom, ak budú vstupy modelov predspracované metódami učenia bez učiteľa. Postup a výsledky tohto experimentovania sú obsahom podkapitoly 6.2. Teória ohľadom použitých metód učenia s učiteľom sa nachádza v kapitole 3.

Kapitola 2

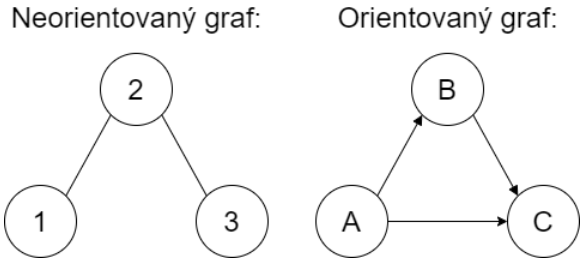
Úvod do dolovania znalostí z grafov

Obsahom prvej podkapitoly je vysvetlenie nasledujúcich pojmov: graf, podgraf, vlastnosti grafu. Vlastnosti grafov sú dôležité pre podkapitolu 5.1, kde je pomocou nich zadefinovaný behaviorálny graf. V prvej podkapitole je ďalej vysvetlené, čo znamená prechádzka cez graf a sú tu popísané všetky maticové reprezentácie grafov, ktoré sú využívané metódami z kapitol 3 a 4. Princíp fungovania Dijkstrovho algoritmu, využitého pri spájaní podgrafov (viď podkapitola 6.2), je tu vysvetlený taktiež. V druhej podkapitole sú vypísané a v krátkosti okomentované základné typy dolovacích úloh nad grafmi. Každá z uvedených typov dolovacích úloh sa v práci riešila, či už v menšej (predikcia hrán) alebo väčšej miere (detekcia komunít).

2.1 Graf

Graf G je formálne možné zapísať pomocou dvojice: $G = (U, H)$. U je označenie pre množinu všetkých uzlov v grafe. V obrázku 2.1, pre graf naľavo $U = \{1, 2, 3\}$ a pre graf napravo $U = \{A, B, C\}$. Z obrázku je vidieť, že uzly v grafoch môžu mať identifikátory rôznych dátových typov (celé čísla, znaky, reťazce, ...). H je označenie pre množinu hrán v grafe. Prvkami H môžu byť dvojprvkové množiny: $\{uzol1, uzol2\}$ alebo usporiadané dvojice: $(uzol1, uzol2)$. Graf, ktorého H obsahuje dvojprvkové množiny sa nazýva **neorientovaný graf**. Príklad takéhoto grafu je zobrazený na obrázku 2.1 vľavo ($H = \{\{1, 2\}, \{2, 3\}\}$). Graf, ktorého H obsahuje usporiadané dvojice sa nazýva **orientovaný graf**. Prvá zložka dvojice ($uzol1$) predstavuje uzol, z ktorého hrana vychádza a druhá zložka dvojice ($uzol2$) prestavuje uzol, do ktorého hrana smeruje. Príklad takéhoto grafu je zobrazený na obrázku 2.1 vpravo ($H = \{(A, B), (B, C), (A, C)\}$). Ak majú hrany v grafe pridelené aj váhy (váhy môžu predstavovať, napr. vzdialenosti medzi uzlami), potom sa takýto graf nazýva **ohodnotený graf**.

Uzly aj hrany v grafoch môžu mať atribúty rôznych dátových typov držiace užitočné dáta. Pre lepšiu predstavu využitia atribútov v grafoch a pochopeniu ďalšej vlastnosti grafov si predstavme nasledujúcu situáciu. Máme graf, ktorého uzly v grafe reprezentujú osoby a hrany priateľstvo medzi osobami. V takomto prípade môže byť atribútom uzlov, napríklad meno alebo pohlavie osoby a atribútom hrany môže byť dĺžka priateľstva. Takýto graf má iba jeden typ uzlov: uzly reprezentujúce osoby. Grafy iba s jedným typom uzlov nazývame **homogénne grafy**. Ak by sme do predstaveného grafu pridali uzly reprezentujúce autá (a napríklad aj hrany reprezentujúce vzťah: vlastník auta), tak by sa v grafe nachádzali už 2



Obr. 2.1: Príklad orientovaného a neorientovaného grafu

typy uzlov. Grafy s dva a viac uzlami sa nazývajú **heterogénne grafy**. Ďalším dôležitými pojmami pre túto prácu a aj obecne v teórií grafov sú nasledujúce pojmy:

- Graf $G = (U, H)$ je **podgrafom** grafu $G' = (U', H')$ v prípade, ak platí $U \subseteq U'$ a $H \subseteq H'$ a existuje izomorfizmus medzi G a G', t.j. bijektivné zobrazenie $f : U \to U'$, pri ktorom platí, že každé 2 uzly $u, v \in U$ sú susedné v grafe G, iba ak uzly $f(u), f(v) \in U'$ sú susedné v grafe G'.

- **Stupeň uzla** (v teórii grafov ide o funkciu označovanú ako $deg(uzol)$) predstavuje v neorientovanom grafe počet hrán zasahujúcich do daného uzla. Pre neorientovaný graf na obrázku 2.1: $deg(1) = 1$, $deg(2) = 2$, $deg(3) = 1$. V prípade orientovaných grafov môže byť stupeň uzla rozdelený ešte na vstupný ($deg^+(u)$) a výstupný ($deg^-(u)$) stupeň uzla. Vstupný stupeň uzla predstavuje počet hrán vstupujúcich do daného uzla a výstupný stupeň uzla predstavuje počet hrán vychádzajúcich z daného uzla. Pre orientovaný graf na obrázku 2.1: $deg^+(a) = 0$, $deg^-(a) = 2$, $deg^+(b) = 1$, $deg^-(b) = 1$, $deg^+(c) = 2$, $deg^-(c) = 0$.

- **Prechádzka nad grafom** znamená budovanie postupnosti uzlov pri nadväzujúcich prechodoch cez hrany v grafe. Prechádzka musí mať definovaný uzol, z ktorého prechádzka začína. Taktež musí mať definované, po koľkých prechodoch má byť prechádzka ukončená, pretože sa uzly v budovanej postupnosti môžu opakovať a prechádzka musí byť v určitom bode ukončená.

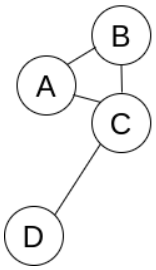
- **Klika** je taký podgraf nejakého grafu, ktorého každý z n uzlov je prepojený hranami s ostatnými $n-1$ uzlami tohto podgrafu (graf s takouto vlastnosťou sa nazýva **úplný graf**).

- **Motív** grafu je vzor (podgraf s minimálne 3 uzlami), ktorý sa často opakuje v nejakom grafe.

Existuje viacero spôsobov ako reprezentovať grafy v počítačoch. Pri dolovaní znalostí z nich, pomocou metód strojového učenia, sa väčšinou využívajú reprezentácie grafov vyjadrené maticami [16]. Jednou z matíc, ktorou sa dá reprezentovať graf je **matica so stupňami uzlov**. Táto matica je diagonálna a na diagonále má vždy stupeň jedného z uzlov grafu. Ďalšou maticou, ktorou sa dá reprezentovať graf je **matica susedných uzlov**. Riadky a stĺpce matice predstavujú, rovnako ako v matici so stupňami uzlov, všetky uzly grafu. Ak v grafe existuje hrana medzi uzlami, tak hodnota bunky matice má hodnotu 1 (v prípade ohodnotených grafov je tam namiesto hodnoty 1 váha hrany), ináč je tam hodnota 0. Matica je symetrická v prípade, ak reprezentuje neorientovaný graf. Poslednou maticovou reprezentáciou, ktorú si v tomto odseku predstavíme je **Laplaceova matica**. Laplaceova

matica vznikne, ak od matice so stupňami uzlov odčítame maticu susedných uzlov. Táto matica sa v sebe udržiava najviac informácií o štruktúre grafu. Matica zachováva informácie o uzloch (stupne uzlov) a taktiež zachováva vzťahy (hrany) medzi uzlami. Na obrázku 2.2 je zobrazený príklad so všetkými popísanými maticovými reprezentáciami pre graf G. Pre lepší prehľad, aby bolo vidieť, ktoré hodnoty patria, akým uzlom, boli do matíc pridané záhlavia a stĺpec navyše, tým pádom vznikla tabuľková a nie maticová reprezentácia grafov.

Graf G:



Tabuľka so stupňami uzlov:

	A	B	C	D
A	2	0	0	0
B	0	2	0	0
C	0	0	3	0
D	0	0	0	1

Tabuľka susedných uzlov:

	A	B	C	D
A	0	1	1	0
B	1	0	1	0
C	1	1	0	1
D	0	0	1	0

Laplaceova tabuľka:

	A	B	C	D
A	2	-1	-1	0
B	-1	2	-1	0
C	-1	-1	3	-1
D	0	0	-1	1

Obr. 2.2: Tabuľkové reprezentácie grafu G

Medzi jedny z najznámejších algoritmov v celej oblasti informatiky patrí **Dijkstrov algoritmus**. Tento algoritmus slúži pre hľadanie najkratších ciest medzi zvoleným uzlom a všetkými ostatnými uzlami v grafe. Algoritmus funguje v princípe nasledovne [13]:

1. Vstupom algoritmu je graf (môže mať rôzne vlastnosti: orientovaný, neorientovaný, ohodnotený a pod.) a zvolený uzol (ďalej označovaný ako zdrojový uzol), od ktorého sa budú zisťovať najkratšie cesty s ostatnými uzlami (ďalej označované ako cieľové uzly).

2. Algoritmus si udržiava hodnoty všetkých vzdialeností medzi zdrojovým uzlom a cieľovými uzlami. Pri zahájení algoritmu sú vzdialenosti nastavené na najväčšiu možnú hodnotu, teda nekonečno. V prípade nájdenia kratšej cesty medzi zdrojovým uzlom a cieľovými uzlami sú hodnoty vzdialeností, v priebehu algoritmu, aktualizované.

3. Po nájdení najkratšej cesty medzi zdrojovým uzlom a niektorým z cieľových uzlov je uzol pridaný do zoznamu navštívených uzlov, Tento uzol je taktiež pridaný do budovanej najkratšej cesty, ktorá je výstupom algoritmu.

4. Algoritmus končí, ak budovaná cesta zahrňuje všetky uzly v grafe. Výstupom algoritmu je cesta, ktorá spojuje najkratšou vzdialenosťou v grafe, zdrojový uzol so všetkými cieľovými uzlami.

2.2 Typy dolovacích úloh nad grafmi

Dolovanie znalostí z grafov je možné rozdeliť do štyroch základných kategórií [5]:

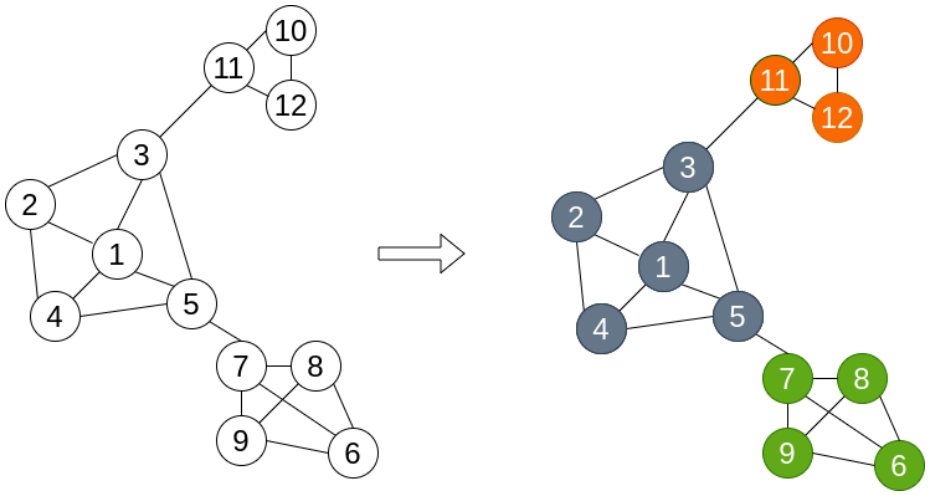
- **Klasifikácia uzlov** je úloha, v ktorej ide o snahu sa z hodnôt atribútov uzlov v grafe naučiť predikovať chýbajúce hodnoty atribútov uzlov toho istého grafu.

- **Predikcia hrán** je úloha, v ktorej ide o hľadanie hrán, ktoré v grafe chýbajú. Napríklad takáto úloha prebieha často na serveroch sociálnych sietí, kde je pre každého

používateľa sociálnej siete generovaný zoznam odporúčaných ľudí pre vytvorenie priateľstva. Sociálna sieť je na serveroch reprezentovaná ako graf, kde ľudia predstavujú uzly a hrany vyjadrujú vzťah priateľstva (v grafe sa nachádzajú určite ešte aj iné typy uzlov a hrán, napríklad uzol: udalosť a hrana: zúčastnil sa, ale tie nás v tejto chvíli nezaujímajú). Zoznam odporúčaných ľudí pre vytvorenie priateľstva sa generuje predikovaním hrán, ktoré v grafe reprezentujúcom sociálnu sieť chýbajú.

- **Klasifikácia grafov** je znova klasifikačná úloha. Cieľom úlohy je z množiny dostupných grafov s štítkami (angl. labels) vytvoriť model a tento model potom použiť pre predikovanie štítkov iných, v priebehu trénovania nevidených, grafov. Táto úloha sa v praxi vykonáva, napríklad s grafmi reprezentujúcimi chemické zlúčeniny. Novej chemickej zlúčenine, reprezentovanej grafom, je pomocou natrénovaného modelu pridelený jeden z nasledujúcich štítkov: zdraviu škodlivá, zdraviu neškodná.

- **Detekcia komunít** je jediná úloha zo zoznamu, ktorá má charakter metódy učenia bez učiteľa (t.j. nové znalosti vznikajú iba z neoznačkovaných dát). Účelom úlohy je, iba na základe štruktúry grafu, vyhľadať v grafe komunity (skupiny) uzlov a to tak, aby: uzly v rámci jednej komunity mali medzi sebou, čo najviac hrán a medzi detegovanými komunitami bolo, čo najmenej hrán. V prípade, ak sa uzol môže nachádzať vo viacerých komunitách, potom ide o **prekrývajúcu detekciu komunít**. V opačnom prípade, ak sa uzol musí nachádzať iba v jednej komunite potom hovoríme o **neprekrývajúcej detekcii komunít**. Príklad neprekrývajúcej detekcie komunít je zobrazený na obrázku 2.3. Pre zopakovanie pojmov: oranžová a zelená komunita uzlov, na uvedenom obrázku, predstavujú kliku.



Obr. 2.3: Príklad neprekrývajúcej detekcie komunít (farby odlišujú nájdené komunity uzlov)

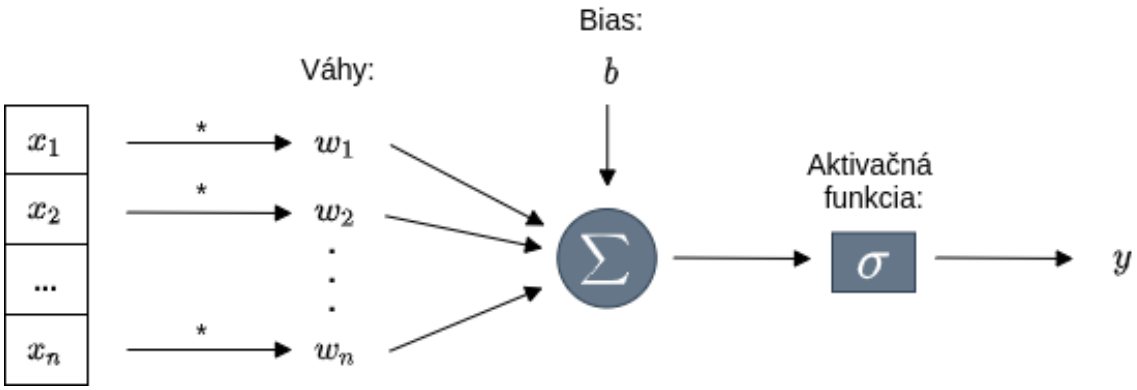
Kapitola 3

Metódy učenia s učiteľom

Kapitola vysvetľuje metódy učenia s učiteľom použité pri experimentovaní opísanom v kapitole 6.2. Pre experimentovanie boli použité iba modely hlbokého učenia, pretože majú schopnosť sa samy naučiť extrahovať zo vstupných dát iba podstatné informácie pre úlohu. Nie je nutné potom prevádzať výber vstupných atribútov modelu manuálne, čož je veľkou výhodou a jedným z dôvodov, prečo sú dnes modely hlbokého učenia, tak populárne.

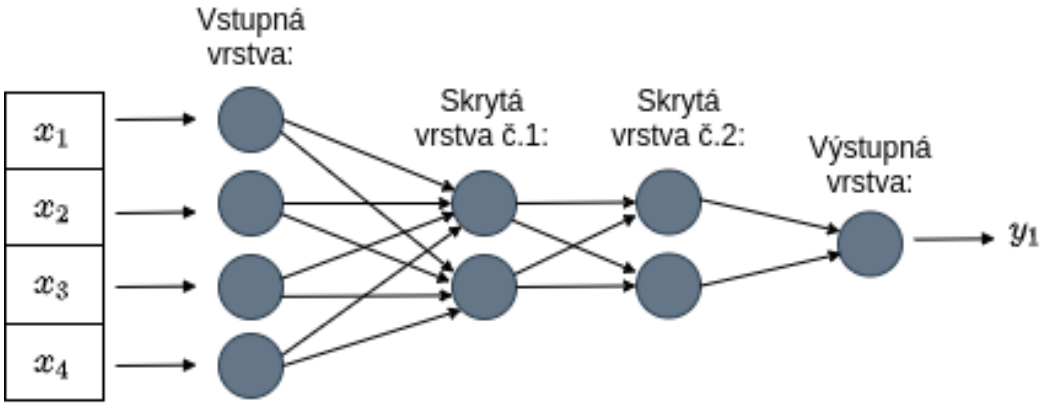
3.1 Dopredné neurónové siete

Dopredné neurónové siete (a všetky ostatné typy neurónových sietí: rekurentné, grafové, ...) predstavujú funkčné aproximátory: Vstupom každej siete je niekoľkorozmerný vektor na základe, ktorého sieť, aproximujúca netriviálnu matematickú funkciu, vyprodukuje výstupný vektor (alebo skalárnu hodnotu). Základnými stavebnými blokmi, z ktorých sú siete zložené, sa nazývajú **neuróny**. Každý neurón je možné zadefinovať trojicou: (vektor váh W, bias b, aktivačná funkcia σ). Fungovanie jedného neurónu je približené na obrázku 3.1. Vstupom neurónu je, rovnako ako u celej neurónovej siete, vektor. Jeho výstupom, však nikdy nie je vektor ale iba skalárna hodnota. Ako funguje neurón: Vektor na vstupe neurónu je najprv vynásobený s váhami neurónu, k sume výsledkov násobení sa následne pričíta bias a táto hodnota potom smeruje na vstup aktivačnej funkcie. Aktivačnou funkciou môže byť hocijaká nelineárna funkcia. Častou používanou aktivačnou funkciou je $f(x) = max(0, x)$, ktorá sa nazýva ReLU (angl. Rectified Linear Unit). Výstup aktivačnej funkcie prestavuje výstup celého neurónu.



Obr. 3.1: Základný budovací blok neurónových sietí – neurón

Väčšou stavebnou jednotkou sietí ako neurón sú **vrstvy** zložené z neurónov. Vrstvy dopredných neurónových sietí sú za sebou zapojené lineárne. Na obrázku 3.2 je možné vidieť príklad doprednej siete s štyrmi vrstvami. Prvá (vstupná) vrstva je špeciálna, pretože počet neurónov v nej musí odpovedať veľkosti vstupného vektora. Skrytých vrstiev môže byť niekoľko (zvyčajne sa v praxi používajú dve vrstvy) a rovnako aj počet neurónov v nich je voliteľný. Počet neurónov vo výstupnej vrstve musí odpovedať počtu hodnôt, ktoré chceme mať na výstupe modelu – aproximovanej funkcie. Sieť zobrazená na obrázku 3.2 produkuje jeden výstup, ktorý napríklad postačuje pre binárnu klasfikáciu.



Obr. 3.2: Príklad doprednej neurónovej siete s 2 skrytými vrstvami

Proces, keď sieť z vektorov na vstupe počíta svoje výstupy, sa nazýva **doprednýprechod sieťou**. Výpočetne efektívnou variantou počítania pri doprednom prechode siete je maticové násobenie. Výpočty jednej vrstvy, pre viaceré vstupné vektory uložené v riadkovej matici X, sa dajú realizovať týmto maticovým násobením:

$$X' = \sigma(X\Theta^T) \tag{3.1}$$

kde: X' je riadková matica s vektormi na výstupe vrstvy; X je riadková matica so vstupnými vektormi; Θ je riadková matica s váhami neurónov vo vrstve; σ je zvolená aktivačná funkcia.

Proces, keď sa sieť učí, z dvojíc: (vstupný vektor, správny výstupný vektor – angl. ground truth) ako správne aproximovať funkciu, sa nazýva **spätné šírenie chyby**. Spätné šírenie chyby nastáva vždy až po doprednom prechode a to pri minimalizácií zvolenej chybovej funkcie (angl. loss function). Hodnota chybovej funkcie vyjadruje ako veľmi sa líšia správne výstupné vektory (druhá položka z uvedenej dvojice) a výstupy siete, ak na vstup siete privedieme vstupné vektory (prvá položka uvedenej dvojice). Pri spätnom šírení chyby sú váhy a biasy neurónov upravované pomocou gradientov tak, aby výstup modelu odpovedal správnemu výstupnemu vektoru. Pre pochopenie experimentov vykonaných v práci nie je potrebné zajsť do detailov počítania gradientov a vysvetleniu optimalizačných techník (*SGD*, *Adam*, ...). V ďalšom odseku sú predstavené v práci použité chybové funkcie.

Binary cross entropy (vzorec 3.2) je loss funkcia používaná pri učení modelu, ktorý má klasifikovať vstup (reprezentovaný vektorom) do jednej z dvoch tried. **Focal loss** (vzorec 3.3) má pôvod pri spracovaní obrazu. Je veľmi podobná Cross entropii a časť ich vzorcov je úplne rovnaká (viď vo vzorcoch uvedených pod odsekom, je tučným písmom vyznačené, ktoré časti majú obidve funkcie úplne rovnaké). Výhoda Focal loss spočíva v tom, že sa snaží riešiť problém, keď trénovacia dátová sada obsahuje jednu majoritnú a druhú minoritnú triedu. Pri trénovaní s nevyváženou dátovou sadou a Cross entropiou to často dopadne

tak, že model bude po trénovaní klasifikovať svoje vstupy iba do majoritnej triedy. Aby toto nenastalo, Focal loss dopĺňa Cross entropy o dve výrazy, ktoré viac trestajú chybu pri nesprávnom klasifikovaní triedy (hyperparameter γ) a dávajú väčšiu váhu chybám pri klasifikácií minoritnej triedy (hyperparameter α) [3]. Poslednou chybovou funkciou, ktorá sa v práci použila (iba v jednom prípade) je **funkcia strednej kvadratickej chyby** (angl. MSE – Mean Square Error, vzorec 3.4). Táto chybová funkcia sa používa v prípade, ak má model predikovať spojité a nie diskrétne hodnoty. Hodnota tejto funkcie rastie s chybami pri predikcii kvadraticky, čo môže byť problém, ak dataset obsahuje nejaké anomálie (chybne anotované správne výstupy). Vzorce všetkých opisovaných chybových funkcií:

$$BCE = -\frac{1}{n}\sum_{i=1}^{n}(\hat{\mathbf{y}}_\mathbf{i}\,\mathbf{ln}(\mathbf{y_i}) + (\mathbf{1}-\hat{\mathbf{y}}_\mathbf{i})\,\mathbf{ln}(\mathbf{1}-\mathbf{y_i})) \tag{3.2}$$

$$FL = -\frac{1}{n}\sum_{i=1}^{n}(\alpha\,(1-y_i)^\gamma\,\hat{\mathbf{y}}_\mathbf{i}\mathbf{ln}(\mathbf{y_i}) + (1-\alpha)\,(1-(1-y_i))^\gamma\,(\mathbf{1}-\hat{\mathbf{y}}_\mathbf{i})\,\mathbf{ln}(\mathbf{1}-\mathbf{y_i})) \tag{3.3}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.4}$$

kde vo všetkých vzorcoch chybových funkcií: n značí počet trénovacích dvojíc; $\hat{y}_i$ značí správnu triedu; y_i hodnotu na výstupe modelu; α a γ sú parametre Focal loss.

Po natrénovaní akéhokoľvek modelu je model potrebné otestovať a ohodnotiť jeho kvalitu. Pre ohodnotenie kvality modelov učenia s učiteľom boli v práci použité tieto tri metriky:

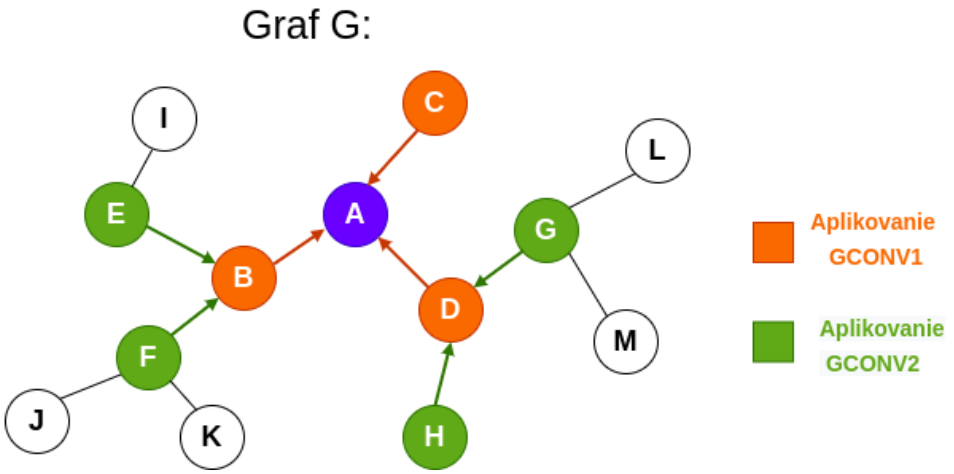
- **Presnosť** (angl. accuracy) je percento správnych predikcií modelu voči všetkým vykonaným predikciám.
- **AUC** (angl. Area Under Curve) je plocha pod ROC (angl. Receiver Operating Characteristic) krivkou a vyjadruje v intervale $< 0,1 >$ schopnosť modelu rozlišovať medzi dvomi predikovanými triedami pri rôznom nastavením prahov pre klasifikáciu (aj iných prahov než hodnota je $0,5$). Čím je hodnota AUC väčšia, tým vie model pri testovaní lepšie rozlišovať medzi triedami.
- **FNR** (angl. False Negative Rate) je metrika, ktorá je pre klasifikáciu do tried: čistý prvok (negatívna trieda) / malware prvok (pozitívna trieda), veľmi dôležitá, keďže ju chceme mať v praxi, čo najnižšiu. Chceme, čo najmenej prípadov, keď model klasifikuje prvok do triedy čistých prvkov a v skutočnosti ide o malware prvok. Táto metrika sa počíta ako pomer nepravdivých predikcií negatívnej triedy (angl. false negatives) so súčtom pravdivých predikcií pozitívnej triedy (angl. true positives) a nepravdivých predikcií negatívnej triedy (angl. false negatives).

3.2 Grafové neurónové siete

Grafové neurónové siete sú teoreticky ideálnym modelom pre dolovanie znalostí z grafov. Siete sa skladajú z niekoľkých grafových konvolučných operátorov, ktoré okrem matice s vstupnými vektormi (napr. atribúty uzlov) majú možnosť pre svoju úlohu využívať aj informáciu zakódovanú v grafovej štruktúre. Typov grafových konvolučných operátorov sú desiatky a stále pribúdajú, pretože výskum, v oblasti grafových neuronových sietí, je stále

aktivný. Jedným zo zaujímavých typov operátorov je relačný grafový konvolučný operátor, ktorý dokáže, pri dolovaní znalostí, využívať aj informácie o type hrán medzi uzlami.

Učenie grafových neurónových sietí je v princípe totožné ako v prípade dopredných neurónových sieti, t.j. cieľom je pri učení upravovať parametre siete tak, aby sa hodnota loss funkcie minimalizovala. Grafové neurónové siete sú iné v tom ako prebieha dopredný prechod sieťou. Vstupom každého grafového konvolučného operátora je vektorová reprezentácia uzla v grafe (počiatočnou vektorovou reprezentáciou uzlov môžu byť atribúty uzlov) a informácia o celej grafovej štruktúre. Výstupom každého operátora je nová vektorová reprezentácia uzla. Dimenzionalita reprezentácie uzla na výstupe môže byť iná ako na vstupe, ide o voliteľný parameter. Pri použití jedného grafového konvolučného operátora je vektorová reprezentácia uzla na výstupe operátora, ovplyvnená reprezentáciami jeho susedných uzlov a jeho starou reprezentáciou. Tento výpočet je inšpirovaný konvolúciami používanými pri spracovaní obrazu, kde výpočet novej reprezentácie každého pixela v obraze, taktiež závisí od reprezentácií jeho susedov a starej reprezentácie seba samého. Pri zapojení dvoch grafových konvolučných operátorov za sebou je výstupná vektorová reprezentácia uzla ovplyvnená všetkými uzlami v maximálnej vzdialenosti dvoch hrán od daného uzla (viď obrázok 3.3). Obecne, pri zapojení n operátorov je výstupná vektorová reprezentácia uzla ovplyvnená všetkými uzlami v maximálnej vzdialenosti n od daného uzla. V praxi sú však málokedy použité viac ako tri operátory, keďže väčšie okolie uzla ako vo vzdialenosti tri, nepotrebujeme vyšetrovať.



Obr. 3.3: Obrázok zobrazujúci uzly, ktoré sú použité pre výpočet novej vektorovej reprezentácie uzla 'A' pri lineárnom zapojení dvoch grafových konvolučných operátorov označených ako 'GCONV1' a 'GCONV2' nad grafom G

V prípade **základnej verzie grafového konvolučného operátora** je výpočet x'_i – novej vektorovej reprezentácie uzla i, daný ako normalizovaná suma: vektorových reprezentácií jeho susedov a jeho vektorovej reprezentácie na vstupe operátora [11]. Vzorec pre výpočet vyzerá nasledovne (zlomok v sume predstavuje normalizačný člen):

$$x'_i = \sigma(\Theta \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{e_{j,i}}{\sqrt{\hat{d}_j \hat{d}_i}} x_j) \tag{3.5}$$

kde: x_j je vektorová reprezentácia uzla j na vstupe operátora; Θ je matica váh operátora; $\mathcal{N}(i)$ je funkcia vracajúca pre uzol i množinu jeho susedov; $e_{j,i}$ je váha hrany medzi uzlom j a i; $\hat{d}_u$ je časť normalizačného člena a pre uzol u: $\hat{d}_u = 1 + \sum_{k \in \mathcal{N}(u)} e_{k,u}$.

Výpočet x'_i – novej vektorovej reprezentácie uzla i, pomocou **relačného grafového konvolučného operátoru** je odlišný tým, že pred normalizovanou sumou je vykonané násobenie vektorových reprezentácií susedských uzlov s viacerými možnými maticami váh. V tomto operátore, má každý typ hrany (typ hrany je označený ako r) medzi uzlami v grafe, svoju vlastnú inštanciu matice váh (Θ_r) a susedský uzol je vynásobený iba s maticou pre typ hrany, ktorý ho spojuje s uzlom i [15]. Ak sa v grafoch, s ktorými má operátor pracovať nachádzajú stovky typov hrán, pamäť potrebná pre uloženie modelu s týmto typom operátora, môže byť obrovská. Rozdiel od základného grafového konvolučného operátora, je aj v tom, že x_i – stará vektorová reprezentácia uzla i, je spracovaná samostatne od reprezentácií susedských uzlov, t.j. vynásobenie s vlastnou maticou Θ_{root}. Vzorec pre výpočet vyzerá nasledovne:

$$x'_i = \sigma(\Theta_{\text{root}} \cdot x_i + \sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_r(i)} \frac{1}{|\mathcal{N}_r(i)|} \Theta_r \cdot x_j) \tag{3.6}$$

kde: x_j je vektorová reprezentácia uzla j na vstupe operátora; Θ_{root} je matica váh pre staré reprezentácie (x_i) aktuálne počítaných uzlov (i); Θ_r je matica váh pre typ hrany r; $\mathcal{R}$ je množina všetkých typov hrán v skúmaných grafoch; $\mathcal{N}_r(i)$ je funkcia vracajúca pre uzol i množinu jeho susedov po hrane typu r.

Pre obidve vysvetlené varianty grafových konvolučných operátorov existujú aj efektívnejšie maticové formy výpočtov. Pre vysvetlenie ako operátory fungujú, to však bolo rozumnejšie vysvetliť na sekvenčných výpočtoch.

Kapitola 4

Metódy učenia bez učiteľa

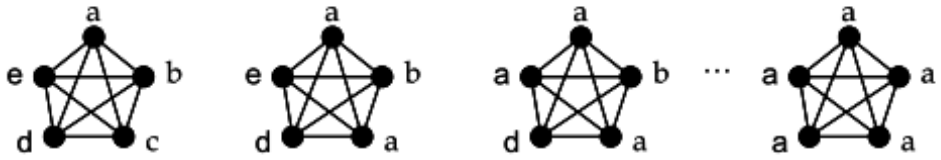
Obsahom tejto kapitoly je vysvetlenie v práci použitých metód učenia bez učiteľa. Použité metódy učenia bez učiteľa je možné rozdeliť do 3 kategórií: metódy pre detekciu komunít (metódy pre hľadanie komunít v grafoch na základe grafovej štruktúry), metódy pre extrakciu vektorových reprezentácií (metódy pre vytváranie vektorových reprezentácií uzlov, v ktorých je zahrnutá informácia o štruktúre uzlov), metódy pre zhlukovanie (klasické zhlukovacie metódy, ktoré nad množinou dátových bodov, vo vysokodimenzionálnom priestore, hľadajú skupiny blízko nachádzajúcich sa bodov).

4.1 Metódy pre detekciu komunít z originálnej grafovej formy

Metódy pre detekciu komunít sú metódy, ktoré hľadajú komunity uzlov iba na základe štruktúry grafu. Pojem komunita nemá v literatúre jednotnú formálnu definíciu. V práci si je pod pojmom komunita grafu možné predstaviť skupinu uzlov grafu, ktoré majú medzi sebou mnoho hrán a s ostatnými uzlami v grafe majú iba málo hrán. Komunity uzlov je v grafoch možné detegovať dvomi spôsobmi. Prvým spôsobom je komunity detegovať v originálnej podobe grafu. To znamená, že uzly z grafov nie je potrebné pre tieto metódy transformovať do vektorov. Komunity sa v grafoch dajú detegovať aj iným spôsobom: Štruktúru uzlov je možné najprv zakódovať pomocou modelov hlbokého učenia do vektorov a následne zavolať zhlukovaciu metódu, ktorá nad vzniknutými vektormi deteguje komunity uzlov. Výhodou detekcie komunít v ich originálnej podobe je to, že sa z grafov nevytratí žiadna informácia, metódy dávajú lepšie výsledky a sú rýchlejšie (niektoré z týchto metód majú dokonca lineárnu časovú zložitosť). Výhodou práce so štruktúrou zakódovanou vo vektoroch je to, že vektory je možné použiť nielen pre detekciu komunít ale aj ako vstup pre iné modely (napr. modely pre klasifikáciu). V tejto podkapitole sú predstavené metódy, ktoré komunity hľadajú v originálnej podobe grafov.

Prvá metóda, ktorá bude v tejto podkapitole predstavená sa nazýva ***LabelPropagation*** [14]. Metóda *Label Propagation* využíva pre detekciu komunít heuristickú funkciu. Metóda začína tým, že každému uzlu z grafu priradí unikátny štítok. Ďalej sa v náhodnom poradí uskutočňujú iterácie cez všetky uzly grafu. V každej iterácií je pomocou heuristickej funkcie náhodne vybranému uzlu priradený nový štítok. Heuristická funkcia tejto metódy funguje podobne ako známa klasifikačná metóda *K-Nearest Neighbors*: vzoru bez štítku je priradený taký štítok, ktorý majú jeho $k-$najbližší susedia najčastejšie. Akurát v tomto prípade sú tu na miesto vzorov uzly a najbližší susedia sa nehľadajú v priestore ale sú pevne daní uzlami, s ktorými má uzol nejaké spoločné hrany. V prípade, ak susedné uzly nejakého uzla majú

viac štítkov, ktoré sa tam nachádzajú najčastejšie, tak je z nich vybraný jeden náhodný štítok. Iterácie cez všetky uzly grafu sa opakujú až do chvíle, keď má každý uzol rovnaký štítok ako má väčšina jeho susedov. Uzly s rovnakými štítkami, po ukončení metódy, predstavujú nájdené komunity uzlov. Celé fungovanie metódy je pre lepšie pochopenie ukázané na obrázku 4.1.



Obr. 4.1: Príklad fungovania metódy *Label Propagation* na klike s 5 uzlami (obrázok prebratý z [14])

Prvá klika zľava, na obrázku 4.1, zobrazuje stavy štítkov po ich počiatočnom priradení. Ako uzol, ktorý štartuje iterácie cez uzly bol náhodne zvolený uzol so štítkom 'c'. Heuristická funkcia vybranému uzlu, zo štyroch najčastejších štítkov jeho susedov (všetky štítky:'a','b','d','e' s počtom jeden), nastavila náhodný štítok z nich a to štítok 'a'. Zmena štítku je zobrazená na obrázku v druhej klike zľava. Ďalej, keďže sa štítok 'a' v klike nachádzal už jednoznačne najčastejšie (t.j. dva-krát, ostatné štítky iba raz), funkcia tento štítok priradila aj ostatným uzlom v klike. Všetky uzly, z kliky zobrazenej na obrázku 4.1, teda podľa metódy *Label Propagation* tvoria jednu komunitu.

Ďalšími metódami, ktoré budú v tejto podkapitole predstavené sú metódy: ***Edge Motif Clustering*** (ďalej už iba *EdMot*) a ***Louvain metóda*** [12, 9]. *EdMot* metóda sama o sebe nepredstavuje metódu pre detekciu komunít. Cieľom tejto metódy je iba upraviť štruktúru grafu tak, aby sa v nej lepšie hľadali komunity inými metódami, napr. *Louivain metódou*. Štruktúra grafu je pomocou metódy *EdMot* upravená nasledujúcim spôsobom [12]:

1. V grafe na vstupe metódy, sú identifikované $k-$najväčšie podgrafy (k je voliteľný parameter), ktorých hrany medzi sebou vytvárajú trojuholníkové motívy.

2. Do identifikovaných $k-$podgrafov sú pridané ďalšie hrany tak, aby sa z podgrafov stala klika. Týmto spôsobom metóda zvýrazňuje miesta v grafe, ktoré by mala hocijaká metóda pre detekciu komunít ľahko označiť za komunitu.

3. Výstupom metódy je teda, pôvodný graf obohatený o hrany, potrebné pre vytvorenie kliky, v identifikovaných podgrafoch.

Jedným z vhodných adeptov pre nájdenie komunít nad grafovou štruktúrou, ohohatenou o nové hrany pomocou metódy *EdMot*, je *Louivain metóda*. Táto metóda je založená na vylepšovaní metriky nazývanej modularita. Metrika modularita bude predstavená v závere podkapitoly. Fungovanie *Louivain metódy* je približené na obrázku 4.2. Kroky z uvedeného obrázka sú okomentované v nasledujúcom číslovanom zozname:

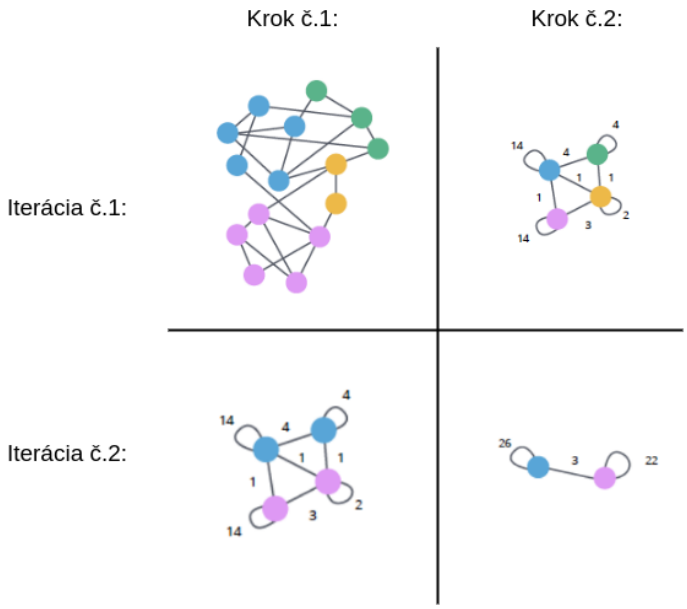
1. Metóda začína tým, že každý uzol v grafe sa nachádza vo vlastnej komunite a váha každej hrany v grafe je nastavená na hodnotu 1.

2. Krokom č.1 je iteratívne prechádzanie cez uzly aktuálnej verzie grafu a postupné spájanie uzlov s komunitami susedných uzlov (na uvedenom obrázku sú komunity vyznačené farbami), ktoré prinesie najlepšie ohodnotenie metrikou modularita. V prípade, ak neexistuje žiadne ďalšie spájanie vedúce k lepšiemu ohodnoteniu metrikou, potom nastáva prechod do kroku č.2.

3. Krok č.2 vykonáva agregáciu uzlov každej nájdenej komunity v kroku č.1 do jedného uzla. Vzniká, tak nový ohodnotený graf, kde váhy na hranách predstavujú sumu váh hrán: vo vnútri komunít (self-loop hrany) alebo medzi komunitami (hrany medzi uzlami).

4. Pokiaľ sa hodnota modularity zlepšuje, tak sa kroky č.1 a č.2 opakujú. V uvedenom obrázku nastali 2 opakovania (iterácie) metódy.

5. Výstupom metódy sú uzly z poslednej vzniknutej verzie grafu, ktoré reprezentujú nájdené komunity. Uzlom z originálneho grafu je priradená komunita na základe toho, do ktorého uzla z finálnej verzie grafu, boli uzly agregované. V grafe na uvedenom obrázku boli nájdené 2 komunity (modrý a ružový uzol).



Obr. 4.2: Príklad fungovania *Louivain metódy* (obrázok upravený z [9])

Záver podkapitoly je venovaný metrikám, ktoré ohodnocujú kvalitu komunít nájdených metódami pre detekciu komunít [6]:

- **Modularita** (angl. modularity) je metrika využívaná ako objektívna funkcia v popisovanej *Louivain metóde*. Táto metrika však slúži, aj ako dobrý spôsob ohodnotenia kvality komunít, nájdených inými metódami. Myšlienka počítania metriky vo vzorci 4.1 je nasledujúca: Pre každú nájdenú komunitu C_k je počítaný rozdiel medzi počtom hrán vo vnútri tejto komunity (prvý člen rozdielu v uvedenom vzorci) a počtom očakávaných hrán vo vnútri komunity (druhý člen rozdielu v uvedenom vzorci). Chceme aby rozdiel bol, čo najväčšia hodnota, t.j. aby skutočný počet hrán v komunite bol

väčší ako jeho očakávaný počet. Výsledná hodnota metriky je spočítaná ako suma (cez všetky nájdené komunity) týchto rozdielov, ktorá je normalizovaná celkovým počtom hrán v grafe ($\frac{1}{2m}$). Hodnota metriky sa takýmto počítaním nachádza vždy v intervale: $< -1, 1 >$. Čím vyššia hodnota, tým lepšie sú najdené komunity.

$$modularita(C) = \frac{1}{2m} \sum_{k=0}^{K} \sum_{i \in C_k} \sum_{j \in C_k} (A_{ij} - \frac{d_i d_j}{2m}) \tag{4.1}$$

kde: $C = \{C_0, C_1, ..., C_{K-1}\}$ je množina nájdených komunít; $K = |C|$; $m = |H|$ (H je množina všetkých hrán v grafe); A_{ij} je prvok matice susedností; d_u je stupeň uzla u.

- **Pokrytie** (angl. coverage) je metrika, ktorú na rozdiel od metriky modularita, zaujímajú nie počty hrán vo vnútri komunít, ale iba to aby medzi komunitami bolo, čo najmenej hrán. Vzorec pre výpočet metriky 4.2, predstavuje jednoduchý pomer medzi počtom hrán vo vnútri všetkých komunít (čitateľ) a všetkými hranami v grafe (menovateľ). Tým pádom, čím je menej hrán medzi komunitami, tým je hodnota v čitateli väčšia a aj hodnota metriky pokrytie je väčšia hodnota. Možný interval pre hodnotu metriky je $< 0, 1 >$. Väčšie číslo predstavuje podľa metriky pokrytie lepšie komunity.

$$pokrytie(G) = \frac{\sum_{k=0}^{K} |(i,j) \in H, i \in C_k, j \in C_k)|}{m} \tag{4.2}$$

kde: $C = \{C_0, C_1, ..., C_{K-1}\}$ je množina nájdených komunít; $K = |C|$; H je množina všetkých hrán v grafe; $m = |H|$.

- **Vodivosť** (angl. conductance) je metrika, ktorú zaujímajú aj počty hrán v rámci komunít, aj počty hrán medzi komunitami. Tým pádom je zo všetkých troch vymenovaných metrík práve táto metrika najkomplexnejšia a podľa definície komunity uvedenej v úvode podkapitoly, najlepšie ohodnocuje kvalitu komunít. Metrika je počítaná pre každú komunitu C_k (viď vzorec 4.3) ako pomer počtu hrán vychádzajúcich z každej komunity s počtom všetkých hrán, ktoré majú uzly vo vnútri komunity. Chceme aby počet hrán vychádzajúcich z komunity bolo, čo najmenšie číslo (čitateľ) a počet hrán uzlov vo vnútri komunity bolo, čo najväčšie číslo (menovateľ). Teda vo výsledku chceme, čo najmenšie číslo ako výsledok pomeru (možný interval pre výsledok pomeru je $< 0, 1 >$). Výpočet vodivosti nad všetkými nájdenými komunitami je uvedený vo vzorci 4.4. Keďže je vo vzorci realizovaný rozdiel čísla 1 od sumy vodivostí cez všetky komunity, chceme potom aby výsledná hodnota vodivosti cez všetky komunity bola, čo najväčšie číslo (možný interval pre výslednú hodnotu vodivosti cez všetky komunity je stále $< 0, 1 >$).

$$vodivost(C_k) = \frac{|(i,j) \in H, i \in C_k, j \notin C_k)|}{min(vol(C_k), vol(\overline{C_k}))} \tag{4.3}$$

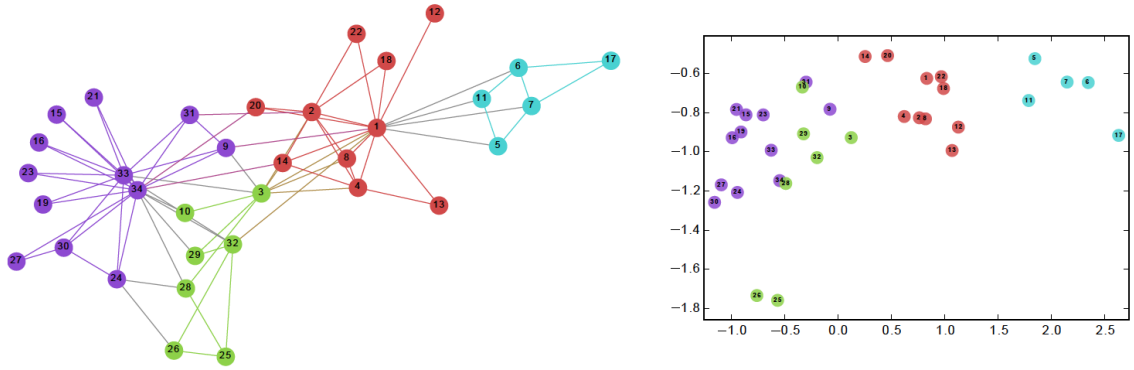
$$vodivost(C) = 1 - \frac{1}{K} \sum_{k=0}^{K} vodivost(C_k) \tag{4.4}$$

kde: $C = \{C_0, C_1, ..., C_{K-1}\}$ je množina nájdených komunít; $K = |C|$; H je množina všetkých hrán v grafe; $vol(C_k) = \sum_{i \in C_k} d_i$; d_i je stupeň uzla i.

4.2 Metódy pre extrakciu vektorových reprezentácií

Najjednoduchším spôsobom ako extrahovať štruktúru uzlov z grafov do vektorov je vybrať vektory: z riadkov Laplaceovej matice alebo z riadkov matice susedných uzlov. Nevýhodou takejto reprezentácie je to, že pre graf s milión uzlami, budú mať všetky extrahované vektory milión dimenzií, ktoré sa nám do operačnej pamäti nemusia vojsť. Ďalšími nevýhodami takýchto vektorov je: riedkosť vektorov (skoro samé nuly vo vektore) a pre každý graf s iným počtom uzlom bude rozmer vektorov (počet dimenzií vo vektoroch) iný. Odlišné rozmery vektorov sú problém, ak by sme nad vektormi chceli aplikovať ľubovoľnú zhlukovaciu alebo klasifikačnú metódu.

Metódy predstavené v tejto podkapitole slúžia pre extrakciu štruktúry uzlov do vektorov, ktorých rozmer je voliteľný parameter metódy. Príklad zakódovania štruktúry uzlov do vektorov je zobrazený na obrázku 4.3: Uzly, ktoré sú v grafe, na uvedenom obrázku, blízko seba (najkratšia cesta medzi uzlami v grafe je malá alebo dokonca majú uzly spoločnú hranu), tak sa v priestore vektorov nachádzajú taktiež veľmi blízko seba. Takýto spôsob zakódovania štruktúry je typický pre metódu *Node2Vec* s použitým DFS spôsobom pre tvorbu trénovacej množiny (existujú aj iné spôsoby zakódovania štruktúry, napr. pomocou grafových autoenkodérov, ktoré budé predstavené v závere kapitoly).

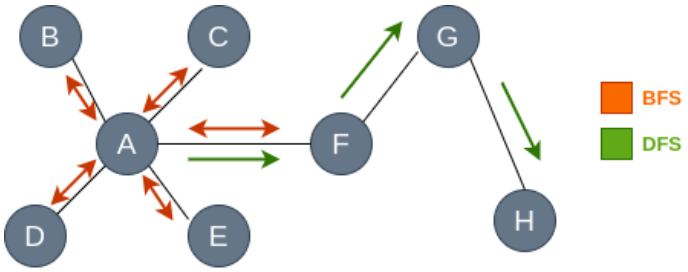


Obr. 4.3: Príklad transformácie štruktúry uzlov grafu do bodov v 2D priestore (prebraté z [7])

Metóda ***Node2Vec*** využívá pre tvorbu vektorov náhodné prechádzky cez graf. V literatúre je možné naraziť ešte na podobnú metódu, ktorá sa nazýva *DeepWalk* [7]. Tieto dve metódy sa od seba líšia sa iba tým, ako vykonávajú prechádzky. Prechádzky sú v metódach použité pre tvorbu trénovacej množiny modelu (trénovanie modelu je popísané v ďalších odsekoch tejto podkapitoly). U metódy *DeepWalk* ide o úplne náhodné prechádzky. U metódy *Node2Vec* sú prechádzky ovplyvnené dvoma parametrami. Parametre udávajú, či sa má jednať o prechádzku v okolí uzla, z ktorého vychádzka začína (pripomína prehľadávanie do šírky – BFS) alebo, či sa má jednať o prechádzku smerujúcu preč od uzla začínajúceho prechádzku (pripomína prehľadávanie do hĺbky – DFS). Na obrázku 4.4 sú zobrazené dve prechádzky cez graf vychádzajúce z rovnakého uzla: 'A'. Oranžovými šípkami je naznačená prechádzka, ktorá zobrazuje prechod uzlami do šírky. Zelenými šípkami je naznačená prechádzka, ktorá zobrazuje prechod uzlami do hĺbky.

Experimenty autorov metódy *Node2Vec* ukázali, že prechádzky do hĺbky sú vhodnejšie, ak nad vzniknutými vektormi tejto metódy chceme, pomocou zhlukovacej metódy, hľadať komunity uzlov [7]. Prechádzky do šírky sa ukázali vhodnejšie, ak nad vzniknutými vektormi

tejto metódy chceme, pomocou zhlukovacej metódy, hľadať skupiny uzlov s približne rovnakými stupňami. Skupina uzlov s približne rovnakými stupňami sa dá interpretovať aj ako skupina uzlov s približne rovnakými úlohami v grafe. Napríklad v grafoch reprezentujúcich počítačové siete, majú smerovače inú úlohu (veľký stupeň uzlov, reprezentujúcich smerovače, v grafe) ako koncové počítače (malý stupeň uzlov, reprezentujúcich počítače, v grafe). Takéto vektory, v ktorých je zakódovaná úloha uzlov v grafe, sú vhodné aj pre trénovanie modelov učenia s učiteľom (viď podkapitola 6.2).



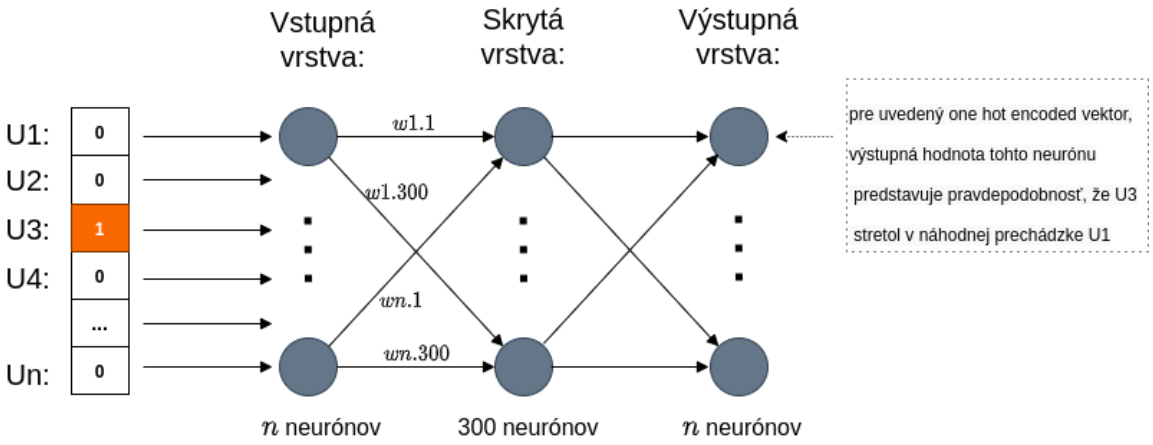
Obr. 4.4: Príklad jednej náhodnej prechádzky cez graf do šírky (vyznačené oranžovými šipkami) a do hĺbky (vyznačené zelenými šípkami)

Princíp fungovania autori metód prebrali z metódy *Word2Vec*. Metóda *Word2Vec* transformuje slová z viet nejakého dokumentu do vektorov. V metódach: *DeepWalk* a *Node2Vec* sa namiesto viet pracuje s postupnosťami uzlov, ktoré vznikli náhodnými prechádzkami cez graf a namiesto slov sa tam pracuje s uzlami. Pre počítanie vektorov popisujúcich uzly sa v metódach používa dopredná neurónová sieť s jednou skrytou vrstvou (viď obrázok 4.5). Počet uzlov grafu, z ktorého tvoríme vektory, si označme ako n. Počet neurónov vo vstupnej a výstupnej vrstve je rovný n. Z takejto štruktúry neurónovej siete už vieme povedať, že počet atribútov každého vzorku použitého pre trénovanie siete sa musí rovnať n. Rovnako zo štruktúry siete vieme povedať, že každý vzor na vstupe bude klasifikovaný do jednej z n tried (výstupom siete bude n pravdepodobností pre každú triedu, o tom, že tam patrí vstupný vzor). Čo je vstupným vzorom, štítkom vstupného vzoru (trieda, do ktorej vstupný vzor patrí) a čo sa vlastne v neurónovej sieti snažíme predikovať?

Vstupnými vzormi pre trénovanie siete a ich štítkami sú vektory, ktoré okrem jednej hodnoty 1 obsahujú, na ostatných miestach vektora, samé hodnoty 0 (angl. **One Hot Encoded vectors**, ďalej už iba OHE-vektory). Každý uzol v grafe má svoj vlastný OHE-vektor, ktorý ho identifikuje. Prvkami trénovacej množiny sú, teda dvojice: (vzor: OHE-vektor uzla1, štítok: OHE-vektor uzla2). To znamená, že sa na základe jedného uzla na vstupe, snažíme predikovať iný uzol. Sémantika klasifikácie je v tom, že neurónová sieť pre uzol na vstupe, počíta n pravdepodobností pre každý ostatný uzol z grafu o tom, že ho uzol na vstupe stretol v náhodnej prechádzke. Množina trénovacích vzorov je získaná z náhodných prechádzok, respektíve sekvencie uzlov, ktoré pomocou prechádzok vznikli. Napríklad z prechádzky: uzol1 -> uzol2 -> uzol3 -> uzol4, je možné pre uzol: uzol2 a okno: 1 (počet uzlov v sekvencii pred a po danom uzle), vytvoriť nasledujúce 2 prvky do trénovacej možiny: (OHE-vektor uzla2, OHE-vektor uzla1), (OHE-vektor uzla2, OHE-vektor uzla3). Veľkosť okna je dôležitá vlastnosť udávajúca, z akého veľkého okolia uzlov, chceme do vektorov vložiť informácie o grafovej štruktúre.

V grafe sme vykonali náhodné prechádzky, z prechádzok sme vytvorili trénovaciu množinu a natrénovali sme neurónovú sieť. Cieľom metód bolo ale vytvoriť vektory popisujúce uzly z grafu. Kde sú tieto vektory? Čo je ďalej potrebné ešte vypočítať? Ďalej už nie

je potrebné počítať nič. Metóda si za vektory popisujúce uzly vyberá natrénované váhy neurónov zo skrytej vrstvy. Napríklad pre uzol: 'U1', na obrázku 4.5, je vektor zložený z váh vychádzajúcich z prvého neurónu vstupnej vrstvy, t. j. váhy: $(w1.1, w1.2, ..., w1.300)$. Obecne pre uzol: 'UX' to budú váhy: $(wX.1, wX.2, ..., wX.300)$. Počet neurónov v skrytej vrstve, udáva dĺžku vytvorených vektorov. Autori metód vybrali počet 300 za vhodný počet neurónov. Túto hodnotu je možné nastaviť ľubovoľne. Záleží od toho aké veľké chceme vektory.



Obr. 4.5: Architektúra neurónovej siete pre vytváranie vektorových reprezentácií uzlov v metódach *DeepWalk* a *Node2Vec*

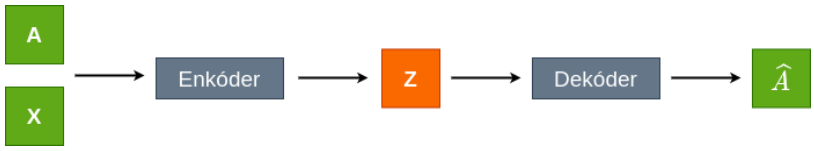
Ďalší spôsob ako vytvoriť vektorové reprezentácie uzlov, ktoré v sebe nesú informáciu o štruktúre uzlov, predstavujú grafové autoenkodéry. V nasledujúcich odsekoch budú predstavené dva typy autoenkodérov: **grafový enkodér** (angl. Graph Auto-Encoder, ďalej označovaný aj ako *GAE*), **variačný grafový enkodér** (angl. Variational Graph Auto-Encoder, ďalej označovaný aj ako *VGAE*) [10]. Fungovanie obidvoch typov grafových autoenkodérov je priblížené na obrázkoch 4.6 a 4.7.

Prvou časťou obidvoch typov autoenkodérov je enkodér. Enkóder sa skladá z grafových konvolučných operátorov zapojených za sebou. Vstupom enkodérov sú: počiatočné vektorové reprezentácie uzlov grafu – matica X (napr. atribúty uzlov) a informácia o štruktúre grafu – matica susedností A. V prípade *GAE* sú, prechodom cez operátory enkodéra, z vstupných matíc: X a A (v matici A je definovaná susednosť uzlov, ktorá je potrebná pre výpočet nových reprezentácií uzlov pomocou grafových konvolučných operátoroch, viď podkapitola 3.2), vytvorené nové reprezentácie uzlov – matica Z. V literatúre sú reprezentácie na výstupe enkodéra nazývané ako **latentné reprezentácie**. V prípade *VGAE* nie je výstupom enkodéru priamo Z, ale niekoľko dvojíc: (μ, σ^2). Zložky dvojice prestavujú parametre normálného rozloženia, ktoré sú použité pre vygenerovanie latentných reprezentácií uzlov – matice Z. Počet dvojíc, ktoré sú na výstupe *VGAE* enkodéru je voliteľný parameter a určuje rozmer latentných vektorov. To, že je do latentných reprezentácií uzlov pridaný šum (generovanie reprezentácií z rozloženia), môže prispieť k lepšej generalizácií autoenkodéra.

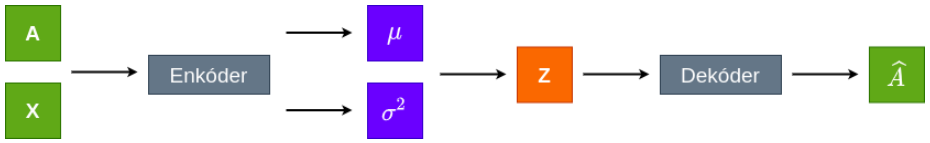
Druhou časťou obidvoch typov autoenkodérov je dekóder. Dekóder, v obidvoch prípadoch, vykonáva rekonštrukciu matice susedností uzlov, prostredníctvom nasledujúceho súčinu matíc: $\sigma(ZZ^T) = \hat{A}$, kde σ je funkcia sigmoida a Z je matica s latentnými reprezentáciami uzlov [10]. Cieľom trénovania je aby sa hodnoty, na rovnakých miestach,

v maticiach A (vstup modelu) a $\hat{A}$ (zrekonštruovaná matica susedností), čo najviac zhodovali. Pre takéto trénovanie sa dá použiť aj obyčajná binárna Cross entropy loss funkcia, ktorá hodnotí ako sa líšia hodnoty na jednotlivých miestach v maticiach A a $\hat{A}$. Pre trénovanie *VGAE* autenkodéru je k rekonštrukčnej chybovej funkcii (rovnaká binary Cross entropy ako u *GAE*) pričítaná ešte hodnota Kullback–Leibler divergencie, ktorá vyjadruje ako sa normálne rozloženia použité pre generovanie latentných reprezentácií uzlov, odlišujú od $\mathcal{N}(0, 1)$ [2]. Hodnota Kullback–Leibler divergencie je do chybovej funkcie pridaná, pretože nechceme aby vo vygenerovaných latentných reprezentáciach boli príliš veľké čísla.

Trénovanie grafových autoenkodérov, teda predstavuje učenie sa vektorové reprezentácie uzlov na vstupe (matica X) pretransformovať tak, aby nad dvoma latentnými reprezentáciami uzlov na výstupe enkodéru (reprezentácie: z_1, z_2), bolo možné nad hodnotou vzniknutou pri skálarnom násobení vektorov ($\sigma(z_1.z_2)$), zistiť, či sa medzi týmito uzlami nachádza hrana (hrana sa podľa autenkodéru v grafe nachádza, ak je hodnota skalárneho súčinu väčšia ako 0). Takýmto spôsobom grafový enkodér tvorí, vektorové reprezentácie uzlov, ktoré v sebe nesú informáciu aj o charakteristikách a aj o štruktúre uzlov. Stále ide o metódu učenia bez učiteľa, keďže nepotrebujeme štítky uzlov, učenie prebieha iba na základe známej grafovej štruktúry.



Obr. 4.6: Princíp fungovania grafového autoenkodéru (inšpirované obrázkom z [8])



Obr. 4.7: Princíp fungovania variačného grafového autoenkodéru (inšpirované obrázkom z [8])

Na záver tejto podkapitoly ešte poznámka, o tom, ako je možné vytvoriť vektorové reprezentácie grafov (prípadne podgrafov) z vektorových reprezentácií uzlov: Vektorové reprezentácie grafov je možne vytvoriť agregovaním všetkých vektorových reprezentácií uzlov, ktoré sa v tomto grafe nachádzajú. Napríklad, ak graf obsahuje 100 uzlov, všetkých jeho 100 vektorových reprezentácií uzlov, je potom agregovaných do 1 vektora. Vhodnými agregačnými funkciami sú: priemer, suma, maximum.

4.3 Zhlukovacie metódy

Metódy z tejto podkapitoly predstavujú klasické zhlukovacie metódy hľadajúce zhluky objektov (napr.: uzlov, grafov, alebo aj osôb) na základe ich podobností. Objekty sú v metódach reprezentované ako $n-$tice. Položky $n-$tice vyjadrujú vlastnosti objektu. Napríklad každú osobu by sme mohli reprezentovať nasledujúcou $n-$ticou: (pohlavie, vek, výška, najvyššie dosiahnuté vzdelanie). Vek a výška sú **kvantitatívne atribúty**. Kvantitatívny atribút nadobúda jednu hodnotu z nekonečného množstva hodnôt. Naproti tomu existujú

ešte **kvalitatívne atribúty**. Kvalitatívny atribút nadobúda hodnotu z konečného množstva hodnôt. Kvalitatívne atribúty sa delia ešte na: **kategorické** (nazývané aj nominálne) a **ordinálne**. Rozdiel medzi nimi je ten, že konečné množstvo hodnôt, ktoré môže ordinálny atribút nadobudnúť, sa dá usporiadať do postupnosti. Z uvedenej $n-$tice, reprezentujúcej osobu, je pohlavie kategorický atribút. Najvyššie dosiahnuté vzdelanie je ordinálny atribút, pretože medzi hodnotami atribútu existuje postupnosť, napríklad: základná škola, stredná škola, vysoká škola.

$N-$tice, reprezentujúce objekty, môžu byť vyznačené vo vysokodimenzionálnych priestoroch ako dátové body. Podobnosť objektov sa potom dá definovať ako vzdialenosť medzi bodmi. Najčastejším typom vzdialenosti, ktorý sa používa pre meranie podobností je **Euklidovská vzdialenosť**. Euklidovská vzdialenosť medzi dvoma bodmi v priestore je definovaná nasledovne: $ED(a,b) = \sqrt{((a_1 - b_1)^2 + ... + (a_n - b_n)^2)}$, kde a, b sú $n-$tice, a_i, b_i sú položky $n-$tice (atribúty objektu) a n je veľkosť $n-$tice. Problém pre takýto výpočet predstavujú kvalitatívne atribúty, ktoré nie sú číselné hodnoty a nedajú sa vložiť do vzorca. Tento problém sa však dá vyriešiť. U kategorických atribútov je možné pre každú hodnotu atribútu pridať do $n-$tice novú položku s možnými hodnotami: 0 a 1. Napríklad pre reprezentáciu osoby $n-$ticou je pre atribút pohlavie osoby do $n-$tice pridať dva binárne atribúty: muž a žena. U ordinálnych atribútov je problém možné vyriešiť prevedením hodnôt ordinálneho atribútu do intervalu: $< 0, 1 >$. Napríklad pre reprezentáciu osoby $n-$ticou je možné najvyššie dosiahnuté vzdelanie (počítame iba s nasledujúcimi rozdelením základná / stredná / vysoká škola) reprezentovať tromi hodnotami: 0, 0.5, 1. Aby kvantitatívne atribúty nemali oproti kvalitatívnym atribútom pre meranie podobností väčší význam pre meranie podobností, tak sa prevádza **normalizácia** ich hodnôt. Normalizácia znamená prevedenie hodnôt atribútu do intervalu: $< -1, 1 >$. Existujú viaceré možnosti ako normalizovať hodnoty atribútov. Najčastejšie používaným spôsobom normalizácie je **štandardizácia**, t. j. prevedenie hodnôt do intervalu $\mathcal{N}(0,1)$. $\mathcal{N}(0,1)$ predstavuje normálne (Gaussovské) rozdelenie pravdepodobnosti so stredom v 0 (používané označenie je: $\mu = 0$) a s rozptylom 1 (používané označenie je: $\sigma^2 = 1$). Previesť hodnoty do takéhoto intervalu je jednoduché. Od každej hodnoty stačí odčítať priemernú hodnotu a následne ju vydeliť smerodajnou odchýlkou.

Vieme už ako reprezentovať objekty pre meranie podobností a ako sa meria podobnosť týchto objektov. Zostáva už iba vysvetliť ako zhlukovacie metódy hľadajú zhluky podobných objektov. Podľa toho ako zhlukovacie metódy fungujú je možné ich rozdeliť do nasledujúcich kategórií:

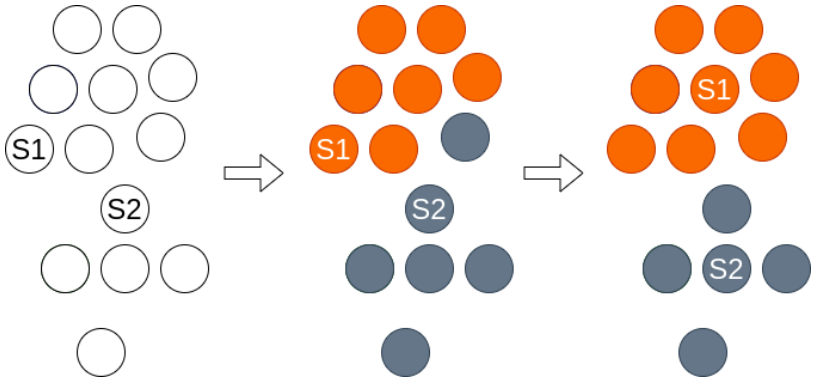
- **Metódy založené na rozdeľovaní**: Metódy rozkladajú priestor objektov do k podpriestorov (k je prirodzené číslo a je nutné ho vhodne zvoliť). Metódou patriacou do tejto kategórie je napríklad metóda: *$K-means$*.

- **Hierarchické metódy**: V metódach sa postupne vytvára hierarchia zhlukov, ktorá začína väčšinou od zhlukov s najpodobnejšími objektmi. O hierarchiu vyššie, spájaním najpodobnejších zhlukov, ďalej vznikajú väčšie zhluky. Spájanie môže končiť až dokiaľ nevznikne jeden zhluk so všetkými objektmi. Existuje aj opačná cesta, teda rozdeľovať zhluky od najväčších zhlukov až po najmenšie (hierarchia od najmenej podobných objektov až po najviac podobné objekty) ale moc sa nevyužíva, pretože ide o výpočetne náročnejšiu variantu. Metódou patriacou do tejto kategórie je napríklad metóda: *BIRCH*.

- **Metódy založené na hustote**: Metódy označujú za zhluk podpriestor, kde je vysoká hustota objektov. Metódou patriacou do tejto kategórie je napríklad metóda: *DBSCAN*.
- **Metódy založené na mriežke**: Priestor objektov je v týchto metódach reprezentovaný mriežkou. Zhluky sú nájdené nad bunkami mriežky. Metódou patriacou do tejto kategórie je napríklad metóda: *WaveCluster*.
- **Metódy založené na modeloch**: Využívajú matematické modely pre popis zhlukov. Metódou patriacou do tejto kategórie je napríklad metóda: *Gaussian mixtures*. Táto metóda sa snaží pre každý zhluk, popísaný normálnym rozložením, nájsť optimálne parametre rozloženia.

V nasledujúcich odsekoch budú bližšie predstavené dve zhlukovacie metódy: najznámejšia zhlukovacia metóda $K-means$ a v práci najviac používaná metóda *MeanShift*. ***K-means*** je metóda založená na rozdeľovaní priestoru na menšie podpriestory. Princíp fungovania metódy je priblížený na obrázku 4.8. Hlavnou nevýhodou tejto metódy je, že používateľ musí na vstupe metódy zadať počet zhlukov, ktoré sa majú v priestore nájsť, t.j. parameter k (na obrázku 4.8 je metóda spustená s $k = 2$). Metóda štartuje tým, že si v priestore vyberie k náhodných objektov reprezentujúcich stredy zhlukov (na obrázku 4.8 sú stredy zhlukov vyznačené ako 'S1' a 'S2') a všetky ostatné objekty v priestore metóda priradí k stredom na základe najmenšej vzdialeností medzi stredmi a objektmi (poznámka: existujú aj iné spôsoby ako inicializovať stredy zhlukov). Ďalej nasleduje cyklus, ktorého cieľom je v jednotlivých iteráciách nájsť lepších reprezentantov stredov zhlukov:

1. V každom zhluku je počítaný priemer nad bodmi, ktoré sa v zhluku nachádzajú. Vypočítané priemery reprezentujú nové stredy zhlukov. Nové stredy zhlukov už nemusia byť reprezentované objektmi, tak ako to je zobrazené na obrázku 4.8 (obrázok predstavuje iba ilustratívny príklad). Môže ísť o hocijaký bod v priestore.
2. Po vypočítaní nových stredov zhlukov sú k stredom zhlukov opäť pridelené ostatné objekty na základe najmenšej vzdialenosti medzi stredmi a objektmi.
3. Tento cyklus sa opakuje až pokiaľ sa členstvo objektov v zhlukoch neprestane meniť (v praxi cyklus končí po stanovenom počte iterácií).



Obr. 4.8: Príklad fungovania metódy *K-means* v 2D priestore so zvoleným $k = 2$

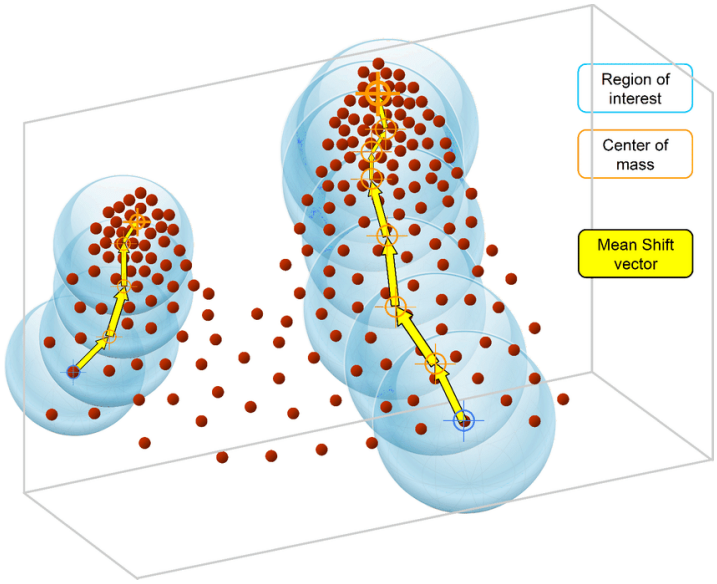
MeanShift je zhlukovacia metóda, ktorá funguje na princípe iteratívneho posúvania dátových bodov po priestore bližšie k oblastiam s najvyššou hustotou bodov. Body sú posúvané po priestore až do chvíle, pokiaľ nenastane konvergencia. Konvergencia v prípade tejto metódy znamená, že dátové body sa pri presúvaní v priestore zredukovali do bodov, ktoré reprezentujú metódou nájdené zhluky. Všetky body z originálneho priestoru, zredukované do jedného z finálnych bodov sú potom označené za jeden zhluk. Nie je, teda nutné na vstupe metódy definovať, koľko zhlukov sa má v priestore nájsť, čož je veľkou výhodou tejto metódy. Nevýhodou metódy je jej časová zložitosť: $\mathcal{O}(n^2)$. Fungovanie zhlukovacej metódy *MeanShift* je približené na obrázku 4.9 a okomentované v nasledujúcom číslovanom zozname:

1. Metóda uskutočnuje iterácie cez všetky dátové body v priestore až do stavu konvergencie. Prvý krokom je výber náhodného dátového bodu, ktorý bude predmetom skúmania v aktuálnej iterácie.

2. Druhým krokom je pre aktuálne skúmaný bod nájsť všetky body v jeho okolí. Okolie boda je dané parametrom metódy: šírka pásma (angl. bandwith). Tento parameter je však možné vhodne odhadnúť pomocou iných metód, ktoré sú v dostupných implementáciach metódy *MeanShift* priamo súčasťou metódy (tým pádom sú tieto implementácie metódy *MeanShift* úplne bezparametrické). Na obrázku 4.9 je okolie skúmaného boda vyznačené modrými kruhmi a pomenované ako oblasť záujmu (angl. region of interest).

3. Tretím krokom je výpočet ťažiska (angl. center of mass) všetkých bodov v oblasti záujmu skúmaného bodu a presunutie skúmaného bodu na miesto ťažiska. Presun bodu na obrázku 4.9 vyznačuje *MeanShift* vektor (angl. *MeanShift* vector).

4. Opakuj kroky č.1, č.2, č.3 až do konvergencie metódy, t.j. pokiaľ je možné aspoň 1 bod v priestor stále presunúť na iné miesto.



Obr. 4.9: Príklad fungovania metódy *MeanShift* (obrázok prebratý z [4])

Kapitola 5

Príprava pred experimentovaním

Obsahom prvej podkapitoly je predstavenie grafov, nad ktorými sa v práci vykonáva dolovanie znalostí. Vypísané sú tu aj všetky úlohy týkajúce sa predprípravy týchto grafov. Opis navrhnutého samoučiaceho systému je obsahom druhej podkapitoly. Tretia podkapitola vysvetľuje fungovanie navrhnutej metriky, ktorá hodnotí kvalitu zhlukov na výstupe navrhnutého samoučiaceho systému.

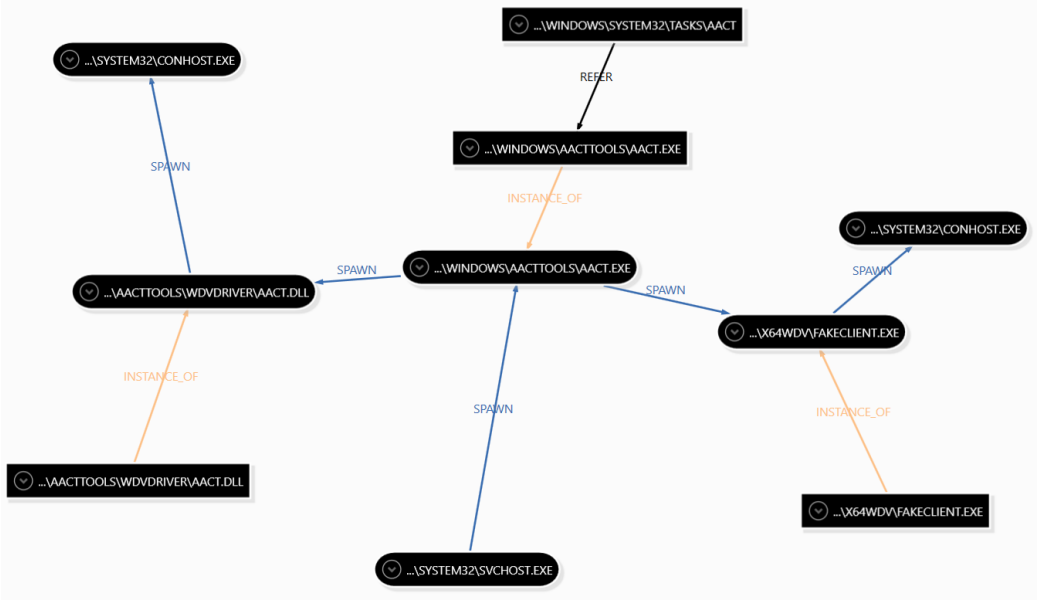
5.1 Behaviorálne grafy

Ako už bolo spomenuté v úvode behaviorálne grafy sú grafy, ktoré zobrazujú chovanie počítačového systému v nejakom zachytenom stave systému. Príklad behaviorálneho grafu je zobrazený na obrázku 5.1. Prvá vlastnosť, ktorá charakterizuje behaviorálne grafy je to, že sú **heterogénne** (uzly sú rôznych typov). Najčastejšími typmi uzlov v grafe sú: proces a vykonateľný súbor. V grafoch sa ešte nachádza aj tretí typ uzlov, nazývaný virtuálny uzol. Virtuálne uzly reprezentujú skripty spustené v príkazovej riadke. Ich výskyt v grafoch je iba raritou. Pre trénovanie modelov učenia s učiteľom a ohodnotenie kvality zhlukov pomocou navrhnutej metriky, boli v práci použité iba uzly typu proces. Je to z dôvodu, že antivírusové programy, v systéme kontrolujú všetky spustené procesy a na základe nich vedia ľahko odhaliť, že sa v systéme nachádza malware. Informácia, o tom, ktorý vykonateľný súbor spustil malware proces, sa dá potom v systéme ľahko dohľadať.

Druhou vlastnosť, ktorou sa tieto grafy vyznačujú je to, že sú **orientované**. Záleží na smere hrán medzi uzlami. Na obrázku 5.1 vpravo dole je proces: '. . . /FAKECLIENT.EXE' inštanciou vykonateľného súboru '. . . /FAKECLIENT.EXE' (vykonateľný súbor — hrana 'INSTANCE OF' —> proces). Opačne to neplatí (súbor nie je inštanciou procesu). To, že sú hrany orientované sa v používaných metódach pre učenie s učiteľom (kapitola 3) a aj bez učiteľa (kapitola 4) príliš nevyužíva, keďže metódy prevažne pracujú s neorientovanými verziami grafov.

Čo však používané metódy vo veľkom využívajú sú atribúty uzlov. Napríklad uzly, typu proces, majú atribúty: PID (identifikátor procesu v systéme) a čas vytvorenia procesu. Atribúty uzlov, ktoré sa v práci využívali pre trénovanie modelov boli však výlučne iba vektory charakteristík. Vektor charakteristík jedného uzla je 163 rozmerný vektor núl a jedničiek, ktorý opisuje vlastnosti daného uzla. Z dôvodu bezpečnostnej politiky firmy Avast nemám možnosť v práci tieto charakteristiky opisovať.

Hrany uzlov v behaviorálnych grafoch **nie sú ohodnotené** (všetky majú rovnakú váhu) ale majú 25 rôznych typov. Na obrázku 5.1 je možné vidieť príklady typov hrán: 'SPAWN',



Obr. 5.1: Príklad behaviorálneho grafu

'INSTANCE OF', 'REFER'. Aj keď názvy typov hrán nie sú chránené politikou Avastu, pripadá mi zbytočne vysvetľovať všetkých 25 typov hrán, keďže jediným v práci použitým modelom, ktorý vedel využiť informáciu o type hrán, bola grafová neurónová sieť s relačnými grafovými konvolučnými operátormi (viď podkapitola 3.2). Najčastejšie sa vyskytujúcimi typmi hrán v grafoch sú: 'INSTANCE OF' a 'SPAWN'. Prvý typ hrany z vymenovaných, v grafe značí, že nejaký vykonateľný súbor spustil proces. Druhý typ hrany značí, že nejaký proces spustil iný proces.

Pre dolovanie znalostí mi bola od Avastu pre prácu poskytnutá sada približne 100000 behaviorálnych grafov vo formáte Google Protocol Buffer[1] (ďalej už iba GPB). Pre použité metódy, implementované v rôznych knižniciach a frameworkoch, bolo nutné grafy previesť do iných formátov. Okrem zmeny formátu grafov prebehlo v čase predprípravy grafov aj ich filtrovanie. Celý **proces predprípravy** vyzeral nasledovne:

1. Grafy z formátu GPB boli prevedené do formátu knižnice NetworkX[2]. Graf prevedený do formátu Networkx predstavuje klasický objekt z objektovo-orientovaného programovania (objekt má nejaké atribúty, napr.: zoznam hrán a uzlov; taktiež nad ním môžeme volať rôzne metódy, napr.: funkciu pre zistenie počtu uzlov v grafe). Formát knižnice NetworkX predstavuje akýsi štandard pre reprezentáciu grafov v obore získavanie znalostí z grafov a tento formát využíva aj knižnica, s metódami učenia bez učiteľa na grafoch, KarateClub[3].

2. Behaviorálny graf nepredstavuje jeden spojitý graf ale skladá sa: z množiny viacerých spojitých grafov, ktoré medzi sebou nemajú spoločné hrany (ďalej sú grafy z množiny nazývané ako grafové komponenty) a sirôt (uzlov bez spojenia s inými uzlami). Z grafov boli vyfiltrované sirotské uzly. Grafové komponenty, s väčším počtom uzlov ako

[1] https://developers.google.com/protocol-buffers
[2] https://networkx.org/
[3] https://karateclub.readthedocs.io/en/latest/

20, boli uložené na disk ako serializované objekty formátu Networkx a ďalej sa s nimi pracovalo ako so samostatnými grafmi. Dôvodom, prečo boli vybrané iba komponenty, s minimálne 20 uzlami, je aby učenie nad nimi bolo pomocou grafových neurónových sietí, čo najviac efektívne: Nové reprezentácie uzlov sú v grafovej neurónovej sieti, počítané hlavne z jeho susedských uzlov. Ak je graf malý, uzol má väčšinou iba jedného suseda a to nevedie na príliš dobré učenie.

3. V práci bol pre dolovanie znalostí z grafov použitý prevažne framework Pytorch[4]. Vstupy, výstupy a parametre Pytorch modelov musia byť uložené vo viacrozmerných maticiach nazývaných tenzory. V práci použité modely si vyžadovali extrakciu nasledujúcich tenzorov: tenzor s charakteristikami uzlov a ich štítkami, tenzor s množinou hrán grafu, tenzor s typmi hrán grafu.

V poslednom bode predspracovania grafov boli spomenuté štítky uzlov. Štítky uzlov (čistý alebo malware) nie sú priamo atribútom uzlov ale je potrebné ich získať pomocou jednej z Avast služieb. Avast služba, Avast REST API Tagger, umožňuje zistiť dodatočné informácie o uzloch typu vykonateľný súbor. Pri HTTP GET požiadavke na službu, kde otlačok (angl. hash) vykonateľného súboru (otlačok je dostupný v atribútoch vykonateľného uzla) je uvedený ako parameter požiadavku, služba vracia informácie o vykonateľnom súbore v JSON formáte. Zo službou vrátenej JSON štruktúry sú pre prácu zaujímavé nasledujúce položky:

- Severity – kategória vykonateľného súboru: malware alebo clean
- Confidence – istota, na koľko percent si je služba istá, že ide o danú kategóriu

To, že pre uzly typu vykonateľný súbor, vieme zistiť hodnoty: severity a confidence, sa v práci využíva pre **štítkovanie uzlov**. Štítkovanie má vždy nastavený prah. Automaticky neznamená, že uzol so severity 'malware', je oštítkovaný triedou malware. Hodnota istoty uzla musí prekročiť nastavený prah, aby uzol mohol byť oštítkovaný ako malware uzol. Rovnako to platí aj pri štítkovaní uzlov so severity 'clean'. Hodnota istoty musí prekročiť prah, aby bol uzol oštítkovaný ako čistý súbor. V práci boli oštítkované iba uzly s istotou väčsou ako 80 %. Uzly, u ktorých bola istota pod týmto prahom neboli oštítkované. Uzol ostane neoštítkovaný aj v prípade, keď Tagger služba o danom uzle nemá uložené potrebné položky: severity a confidence. Uzly teda okrem tried: čistý uzol a malware uzol, môžu patriť ešte do triedy neoznačkovaných uzlov. Pred trénovaním binárnych klasifikačných modelov sú však neoznačkované uzly z trénovacej množiny vyfiltrované.

Informácia o triede uzlov v grafe, je na uzly typu proces, nutné preniesť od uzlov typu vykonateľný súbor, po hranách typu 'INSTANCE OF'. Pre štítkovanie grafov (prípadne podgrafov) bol použitý nasledujúci spôsob: Graf, ktorý obsahuje aspoň jeden uzol, typu proces, oštítkovaný ako malware, je oštítkovaný ako malware graf. V opačnom prípade je graf oštítkovaný ako čistý graf.

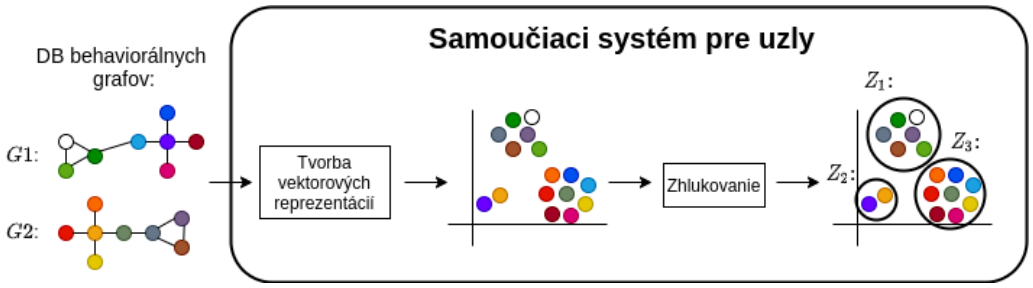
5.2 Návrh samoučiaceho systému

V celej práci prebiehalo dolovanie znalostí z behaviorálnych grafov nad dvoma rôznymi úrovňami ich zloženia: na úrovni celých grafov (prípadne podgrafov) a na úrovni uzlov. Preto

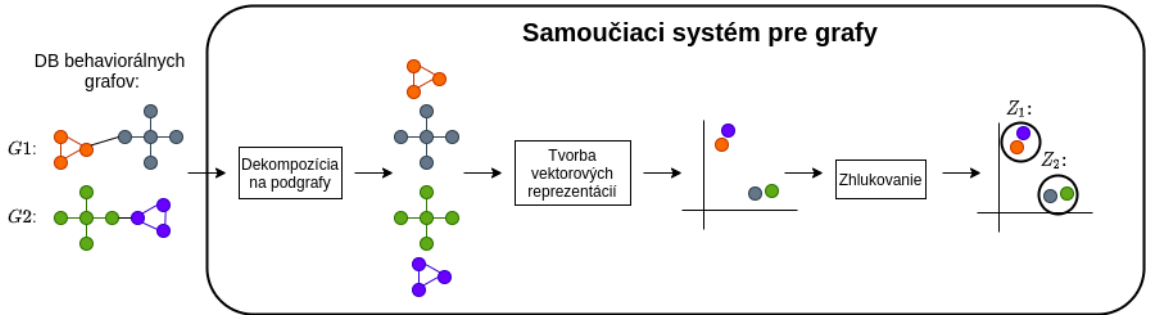
[4] https://pytorch.org/

boli v práci návrhnuté dve samostatné samoučiace systémy: jeden pre dolovanie znalostí z grafov a druhý pre dolovanie znalostí z uzlov. Bloková **schéma systému navrhnutého pre uzly** sa nachádza na obrázku 5.2. **Schéma systému navrhnutého pre grafy** sa nachádza na obrázku 5.3. V oboch prípadoch je cieľom systémov vytvárať zhluky vektorových reprezentácií prvkov, v ktorých sa triedy prvkov (čistý prvok a malware prvok) nemiešajú. Motiváciou vytvárania takýchto zhlukov je to, že zhluk iba s jednou triedou môže byť považovaný za špecifický prípad danej triedy a predstavuje užitočný zdroj ďalších analýz nad behaviorálnymi grafmi.

Rozdiel medzi navrhnutým systémom pre uzly a systémom pre grafy, je v tom, že v prípade grafov je vhodné pred vytváraním ich vektorovej reprezentácie, grafy rozdeliť na menšie podgrafy. Dôvod za rozdelím grafov na menšie časti je v tom, že grafová reprezentácia počítačového systému napadnutého vírusom, nikdy nebude obsahovať iba malware uzly. Aplikovaním metódy pre detekciu komunít nad grafom, chceme separovať časti grafu obsahujúce malware a časti grafu obsahujúce čisté uzly.

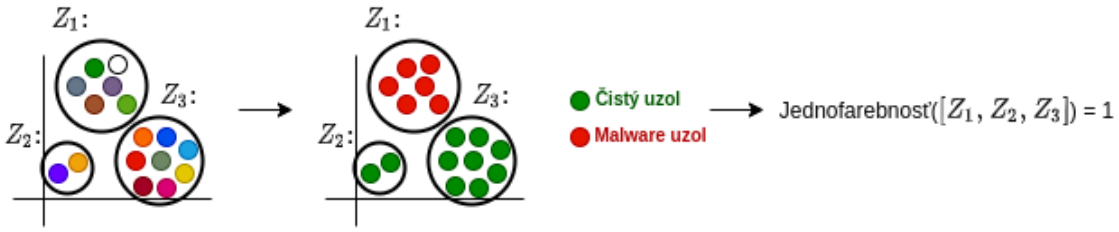


Obr. 5.2: Navrhnutý samoučiaci systém pre uzly behaviorálnych grafov



Obr. 5.3: Navrhnutý samoučiaci systém pre celé grafy

Cieľom experimentovania, popísaného v podkapitole 6.1, bolo vybrať vhodné metódy pre každý funkčný blok z blokových schém navrhnutých systémov (obrázky 5.2 a 5.3) tak, aby bol výstup systému, čo najlepšie ohodnotený pomocou navrhnutej metriky jednofarebnosť. Príklad, ako by mohli byť hodnotené výstupy navrhnutého systému pre uzly – zhluky uzlov, je zobrazený na obrázku 5.4. Vysvetlenie ako sá počíta navrhnutá metrika sa nachádza v nasledujúcej podkapitole.



Obr. 5.4: Ohodnotenie zhlukov, ktoré našiel navrhnutý systém pre uzly behaviorálnych grafov, pomocou navrhnutej metriky jednofarebnosť

5.3 Metrika ohodnocujúca kvalitu zhlukov na výstupe navrhnutého systému

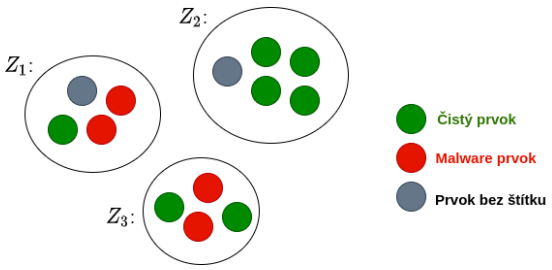
Navrhnutá metrika hodnotí výstup navrhnutého systému na základe zloženia zhlukov, ktoré má systém na výstupe. Konkrétne hodnotí ako dobre sú v zhlukoch oddelené čisté a malware prvky. Navrhnutá metrika je nazvaná **'jednofarebnosť zhlukov'**, pretože vo všetkých obrázkoch v práci, majú čisté prvky zelenú farbu a malware prvky červenú farbu, a v zhlukoch nechceme aby sa tieto dve farby miešali. Pomenovať metriku ako 'čistota zhlukov' bol tiež jeden z návrhov ale mohlo by to byť trochu mätúce pomenovanie metriky, pretože zhluk zložený iba z malware prvkov by potom bol pomenovaný ako 100 percentne čistý zhluk. Vzorec pre výpočet jednofarebnosti nad množinou zhlukov Z:

$$jednofarebnost(Z) = \frac{1}{n}\sum_{i=0}^{n}\frac{max(ciste(Z_i), malware(Z_i))}{ciste(Z_i) + malware(Z_i)} \tag{5.1}$$

kde: Z je množina zhlukov na výstupe systému; $n = |Z|$; $Z_i \in Z$ je zhluk prvkov, v ktorom sa nachádza aspoň jeden oštítkovaný prvok; $ciste(Z_i)$ je funkcia, ktorá vracia počet čistých prvkov v zhluku Z_i; $malware(Z_i)$ je funkcia, ktorá vracia počet malware prvkov v zhluku Z_i.

Hodnota metriky sa pohybuje v rozsahu $< \frac{1}{2}, 1 >$. Čím viac sa hodnota metriky približuje k číslu 1, tým lepšie sú vo vyšetrovaných zhlukoch oddelené čisté a malware prvky (v zhlukoch je viac "jednofarebnosti"). Príklad výpočtu metriky nad zhlukmi zobrazenými na obrázku 5.5:

$$jednofarebnost([Z_1, Z_2, Z_3]) = \frac{1}{3}(\frac{2}{3} + \frac{4}{4} + \frac{2}{4}) = 0,72$$



Obr. 5.5: Obrázok slúžiaci pre príkladový výpočet navrhnutej metriky jednofarebnosť

Kapitola 6

Experimenty

Prvá podkapitola obsahuje popis vykonaných experimentov s metódami učenia bez učiteľa. Cieľom experimentovania v prvej podkapitole je výber vhodných blokov do navrhnutých samoučiacich systémov (viď navrhnuté systémy na obrázkoch 5.2 a 5.3) tak, aby výstup systému dosahoval, čo najlepšej hodnoty metriky jednofarebnosť (viac o metrike v podkapitole 5.3). V druhej podkapitole sú opísané v práci vykonané experimentovania s metódami učenia s učiteľom. Cieľom experimentovania v druhej podkapitole bolo zistiť, či je možné vylepšiť kvalitu výsledkov modelov učenia s učiteľom (dopredné neurónové siete a grafové neurónové siete), ak budú vstupy modelov predspracované metódami učenia bez učiteľa. Ak nie je spomenuté ináč, tak pri experimentovaní opísanom v prvej aj v druhej podkapitole, bola využitá dátová sada s 14 282 behaviorálnymi grafmi (26 347 grafovými komponentami), ktoré obsahovali spolu 2 779 624 uzlov (priemerne 194.62 uzlov na graf). Pričom rozdelenie grafov na trénovaciu, testovaciu a validačnú sadu bolo štandardné: 70 % grafov išlo na trénovanie, 15 % na testovanie a 15 % na validáciu modelov. Pre optimalizovanie váh v dopredných neurónových sieťach bol použitý algoritmus *Adam*. Pre optimalizovanie váh v grafových neurónových sieťach sa ukázalo, že je lepšie použiť klasický optimalizačný algoritmus *Stochastic Gradient Descent*.

6.1 Výber metód pre navrhnutý samoučiaci systém

Aj keď boli v práci navrhnuté dva samoučiace systémy: jeden pre uzly a jeden pre grafy. Základ obidvoch systémov je však rovnaký, t.j. vytváranie vektorových reprezentácií prvkov a zhlukovanie reprezentácií. Najprv sa pozrieme na časť, ktorú má iba jeden zo systémov. Následne prejdeme na časti, ktoré obsahujú oba systémy.

Navrhnutý systém pre grafy je, oproti systému pre uzly, rozšírený o časť, ktorá dekomponuje grafy na podgrafy. Pre dekompozíciu grafov boli použité metódy s implementáciami z nasledujúcich knižníc:

- KarateClub - *LabelPropagation*, *Edmot + Loivain*, *Node2Vec*
- igraph[1] - *InfoMap*
- scikit-learn[2] - *MeanShift*

[1] https://igraph.org/
[2] https://scikit-learn.org/stable/

Výsledky detekcie komunít nad celým datasetom (14 282 grafov) sú zobrazené v tabuľke 6.1. Stĺpec tabuľky 'Dekomponované grafy' vyjadruje koľko percent grafov bolo po použití metódy v skutočnosti dekomponovaných (nie každá metóda dá vždy na výstup množinu komunít ale dá na výstup iba jednu komunitu – celý graf). Nad vektorovými reprezentáciami vytvorenými metódou *Node2Vec*, bola použitá zhlukovacia metóda *MeanShift*, ktorá vytvorila komunity uzlov. Zhlukovanie *Node2Vec* uzlových reprezentácií som sa snažil trochu vylepšiť: uzly v zhlukoch s menej ako štyrmi uzlami som priradzoval k väčším zhlukom na základe ich blízkosti k stredom väčších zhlukov (väčším zhlukom sa myslí zhluk s aspoň štyrmi uzlami). Moc to však nepomohlo. Jedinou metrikou, ktorou som porazil ostatné metódy je metrika pokrytie. Túto metriku zaujíma iba to, že medzi nájdenými komunitami je, čo najmenej hrán. To koľko hrán je vo vnútri komunít ju vôbec nezaujíma (pre vysvetlenie metrík pre detekciu komunít, viď podkapitola 4.1). Najlepšie hodnoty metrík ohodnocujúcich kvalitu komunít vo výsledku dosiahla metóda ***Edmot + Loivain***, ktorá bola preto vybratá ako metóda systému **pre dekomponovanie vstupných grafov systému**. Naopak najhoršie výsledky dosahovala metóda *LabelPropagation*, pretože produkovala nespojité podgrafy.

Metóda	Priemerne uzlov v komunite	Dekomponované grafy [%]	Pokrytie	Modularita	Vodivosť
Label Propagation	19,420	100	0,010	-0,093	-68,283
EdMot + Louvain	32,387	86,3	0,903	**0,643**	**0,647**
InfoMap	9,933	86,3	0,823	0,600	0,363
Node2Vec (DFS varianta)	69,980	100	**0,970**	0,420	0,540

Obr. 6.1: Výsledky experimentovania s rôznymi metódami detekcie komunít nad celým datasetom

Pre vytváranie vektorových reprezentácií boli v práci použité dva typy grafových autoenkóderov: *GAE* a *VGAE*. Vstupom grafových autoenkodérov sú počiatočné vektorové reprezentácie uzlov – vektory charakeristík. Trénovanie grafových autoenkodérov predstavuje učenie sa vektorové reprezentácie uzlov na vstupe pretransformovať tak, aby pri skalárnom súčine dvoch reprezentácií uzlov na výstupe autoenkodéru bolo možné identifikovať, či sa medzi týmito uzlami nachádza hrana. Takýmto spôsobom grafový enkodér tvorí, vektorové reprezentácie uzlov, ktoré v sebe nesú informáciu aj o charakteristikách a aj o štruktúre uzlov. Stále ide o metódu učenia bez učiteľa, keďže nepotrebujeme štítky uzlov (čistý / malware), učenie prebieha iba na základe známej grafovej štruktúry (pre viac informácií o grafových autoenkodéroch, viď podkapitola 4.2). Výsledky binárnej klasifikácie nad testovacou sadou do tried: je medzi dvomi uzlami hrana / nie je medzi dvomi uzlami hrana, sú pre obidve autoenkodéry zobrazené v tabuľke 6.2. *GAE* zvíťazil v obidvoch metrikách.

Model	Dekóder	Presnosť [%]	AUC
GAE	2x Relačný grafový konvolučný operátor	**98,34**	**98,12**
VGAE	2x Obyčajný grafový konvolučný operátor	90,5	91,07

Obr. 6.2: Výsledky predikcie hrán obidvoch použitých variant grafových autoenkóderov (*GAE* a *VGAE*)

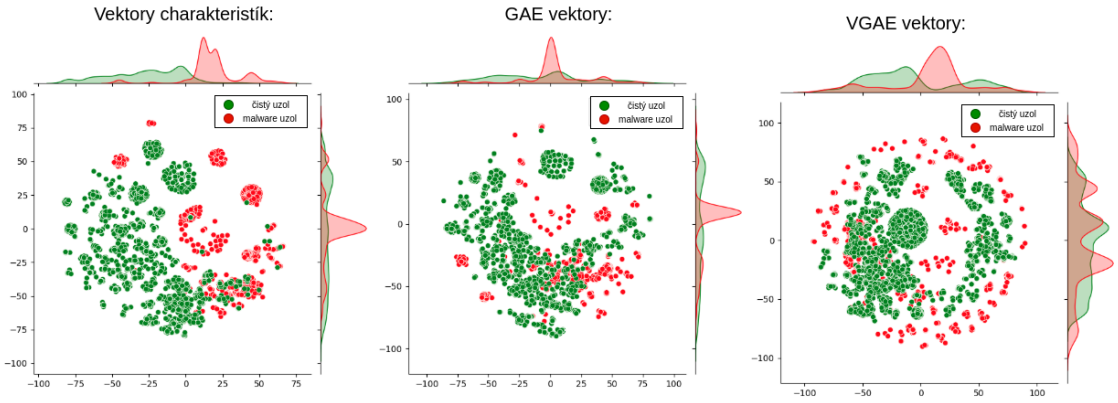
Po natrénovaní autoenkodérov nad trénovacou a validačnou sadou, boli pomocou ich enkodérov vytvorené vektorové reprezentácie uzlov z testovacej sady. Jednou z vektoro-

vých reprezentácií uzlov je aj samotný 163 rozmerný vektor charakteristík, ktorý môže byť taktiež použitý pre hľadanie zhlukov prvkov. Pri experimentovaní nás zaujíma, či reprezentácie na výstupe grafových autoenkodérov, ktoré v sebe nesú informáciu aj o štruktúre grafu, môžu priniesť lepšie výsledky pri hľadaní "jednofarebných"zhlukov ako vektory charakteristík.

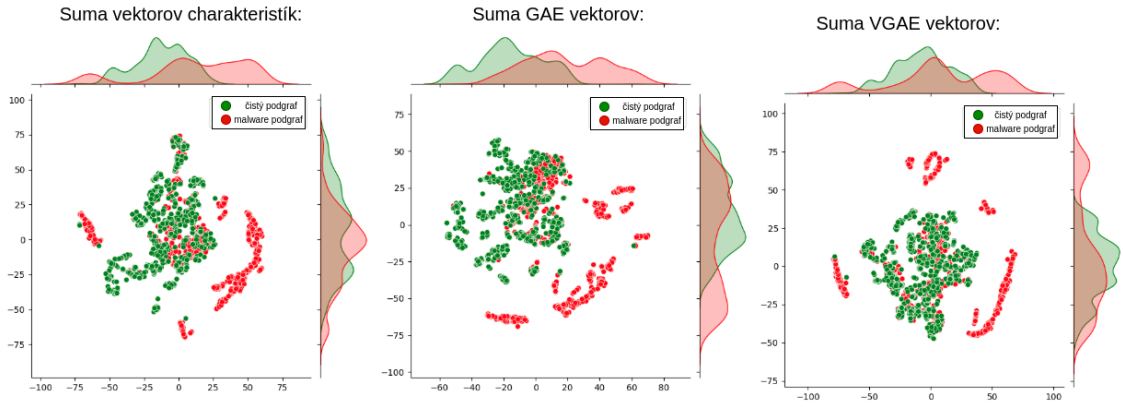
Všetky tri spomínané vektorové reprezentácie prvkov, s vyváženým počtom tried, vytvorené z testovacej sady, redukované do 2D, sú zobrazené na obrázkoch 6.3 (reprezentácie uzlov) a 6.4 (reprezentácie *Edmot + Loivain* podgrafov). Pri tvorbe vektorových reprezentácií podgrafov boli všetky vektorové reprezentácie uzlov podgrafu sčítané do jedného vektora (okrem sumovania vektorov uzlov som skúšal aj priemer vektorov ale suma dávala lepšie výsledky metriky jednofarebnosť). Už z týchto obrázkov je vidieť, ktoré vektorové reprezentácie sú najviac vhodné pre vytváranie "jednofarebných"zhlukov. Pre uzly to sú obyčajné vektory charakteristík a pre grafy reprezentácie vytvorené *VGAE*. Potvrdené je to aj v tabuľke 6.5, kde tieto reprezentácie majú v tabuľke najvyššiu hodnotu metriky jednofarebnosť. **Do navrhnutého samoučiaceho systému pre uzly, sú najlepšou reprezentáciou**, pre vytváranie "jednofarebných"zhlukov, na základe prevedených meraní, **vektory charakeristík a pre navrhnutý samoučiaci systému pre grafy sú najlepšou vektorovou reprezentáciou podgrafov výstupy z variačného grafového autoenkodéru.** Poznámka: 163 rozmerné vektory charakteristík boli pred aplikovaním zhlukovacej metódy najprv redukované do 64 rozmerného vektoru pomocou metódy PCA.

Tabuľka 6.5 okrem jednej hodnoty jednofarebnosti pre všetky zhluky, obsahuje aj jednofarebnosti pre čisté zhluky (zhluk, kde je prevažná väčšina čistých prvkov) a malware zhluky (zhluk, kde je prevažná väčšina malware prvkov). Takýmto rozdelením jednofarebnosti bolo možné zistiť, že napríklad u zhlukovania *GAE* reprezentácií podgrafov, zhlukovacia metóda našla malware zhluky s najlepšou možnou hodnotou jednofarebnosti, t.j. hodnota 1.

Zhluky nad vektorovými reprezentáciami boli v práci vytvárané iba pomocou jednej metódy: *MeanShift*. Výhodou tejto metódy je to, že ide o bezparametrickú metódu. Keďže táto zhlukovacia metóda produkovala po celú dobu experimentovania zmysluplné zhluky, nebolo nutné robiť "konkurz"pre výber zhlukovacej metódy ako v prípade ostatných častí systému. Takže **pre obidve navrhnuté systémy** (aj pre uzly aj pre grafy) **bola metóda *MeanShift* vybratá pre zhlukovanie vektorových reprezentácií** .



Obr. 6.3: Tri rôzne vektorové reprezentácie uzlov (vybratých 9250 zástupcov z testovacej sady pre odidve triedy) zobrazené v 2D priestore (redukcia do 2D bola vykonaná pomocou *TSNE*)



Obr. 6.4: Tri rôzne vektorové reprezentácie *EdMot + Louivain* podgrafov (vybratých 1291 zástupcov z testovacej sady pre odidve triedy) zobrazené v 2D priestore (redukcia do 2D bola vykonaná pomocou *TSNE*)

Prvok	Vstup	Počet zhlukov	Priemerne uzlov	Jednofarebnosť	Počet čistých	Jednofarebnosť čistých	Počet malware	Jednofarebnosť malware
uzol	Vektory charakteristík	231	79,004	**0,963**	122	**0,981**	102	**0,973**
	GAE vektory	144	126,736	0,926	85	0,932	57	0,932
	VGAE vektory	107	170,561	0,914	49	0,918	55	0,933
podgraf	Suma vektorov charakteristík	33	78,242	0,955	22	0,947	11	0,970
	Suma GAE vektorov	50	51,640	0,950	33	0,954	15	**1,000**
	Suma VGAE vektorov	51	50,627	**0,963**	35	**0,970**	15	0,978

Obr. 6.5: Výsledky zhlukovania metódy *MeanShift* nad rôznymi vektorovými reprezentáciami uzlov a podgrafov. Ohodnotenie vhodnosti vektorovej reprezentácie, pre požadované jednotriedne zhlukovanie, je vyjadrené v práci navrhnutou metrikou jednofarebnosť.

6.2 Metódy učenia s učiteľom využívajúce metódy učenia bez učiteľa

Táto podkapitola je rozdelená do troch častí. Prvou časťou sú modely dopredných neurónových sietí natrénované čisto nad vektormi charakteristík. Druhou časťou sú modely grafových neurónových sietí natrénované nad charakeristikami a grafovou štruktúrou. Treťou časťou sú modely dopredných neurónových sietí natrénované nad vektormi, ktoré vznikli pomocou metód učenia bez učiteľa (metódy učenia bez učiteľa sú v tomto prípade použité ako forma predspracovania dát pre trénovanie modelov). Cieľom tohto experimentovania je zistiť, či predspracovanie metódami učenia bez učiteľa (tretia časť) môže priniesť lepšie výsledky klasifikácie ako pri použití modelov, ktoré predspracovanie nevyužívajú (prvá a druhá časť).

V priebehu experimentovania bolo zistené, že modely pre klasifikáciu grafov dávajú lepšie výsledky, ak sú grafy dekompované na podgrafy a až nad podgrafmi je natrénovaný klasifikačný model. Pre dekomponovanie grafov bola použitá metóda *EdMot + Louivain*, u ktorej bolo zistené, že podľa metrík pre detekciu komunít tvorí najlepšie komunity (viď predchádzajúca podkapitola). Ešte lepšie výsledky model pre klasifikáciu grafov dosahoval, ak bol natrénovaný nad podgrafmi, ktoré boli po dekompozícii originálneho grafu spojené s ostatnými podgrafmi, a to v prípade, ak ich malware uzly boli prepojené hranou v originálnom behaviorálnom grafe. Pre hľadanie hrán medzi malware uzlami v originálnom grafe

bol použitý *Dijkstrov algoritmus* (ako algoritmus funguje, viď podkapitola 2.1). Myšlienkou za spájaním podgrafov je to, že podgraf, ktorý vznikol spojením viacerých malware podgrafov, by mal mať v sebe viac informácie potrebnej pre klasifikovanie podgrafu do triedy malware ako samostatné podgrafy, z ktorých spojený podgraf vznikol. Preto sa v tabuľkách zobrazujúcich výsledky modelov nachádza stĺpec 'Prvok', ktorý by štandardne mal obsahovať iba dve položky: uzol a graf, ale vďaka týmto vylepšeniam, uvedené tabuľky v stĺpci 'Prvok' naviac ešte obsahujú položky: podgraf a podgraf po spájaní.

Výsledky klasifikácie dopredných neurónových sietí natrénovanými nad charakeristikami je možné vidieť v tabuľke 6.6. Hneď pri prvých výsledkoch experimentovania s modelmi učenia s učiteľom, sa ukázalo, že lepšie výsledky model dáva, ak bola pre trénovanie použitá Focal loss funkcia namiesto Cross entropie (viac o loss funkciách sa nachádza v kapitole 3.1). Ako je vidieť v uvedenej tabuľke, pri použití Cross entropie bola metrika FNR oproti použitiu Focal loss skoro štvornásobne vyššia, čo znamená štvornásobne viac prípadov, že model označil malware prvok za čistý prvok. Preto bola ďalej pri experimentovaní použitá iba Focal loss. Hyperparametre funkcie boli nastavené nasledovne: $\gamma = 2$ a α bola vypočítaná ako pomer čistých a malware prvkov v datasete. V prípade uzlov: $\alpha = 69,34$ (čo znamená, že na 1 malware uzol v grafoch v datasete pripadá $69,34$ čistých uzlov), v prípade grafov: $\alpha = 3,177$, v prípade podgrafov: $\alpha = 21,092$, v prípade spojených podgrafov: $\alpha = 24,84$). Pri klasifikácií grafov, to dopadlo tak, ako bolo očakávané. Najlepšie výsledky dosiahol model, ak bol natrénovaný nad spojenými podgrafmi.

Prvok	Loss funkcia	Presnosť [%]	FNR [%]	AUC
uzol	Cross entropy	**99,825**	7,642	0,961
	Focal loss	99,021	**1,956**	**0,985**
graf	Focal loss	62,535	**13,122**	0,709
podgraf	Focal loss	89,817	16,110	0,870
podgraf po spájaní	Focal loss	**92,030**	14,236	**0,890**

Obr. 6.6: Výsledky klasifikácie dopredných neurónových sietí s dvomi skrytými vrstvami, natrénovanými čisto na charakteristikách uzlov (v prípade klasifikácie grafov a podgrafov boli charakteristiky uzlov grafu, pred vstupom do modelu, spriemerované).

Výsledky klasifikácie grafových neurónových sietí sú zobrazené v tabuľke 6.7. Typy použitých grafových konvolučných operátorov, ktoré jednotlivé natrénované modely využívali sa nachádzajú v stĺpci tabuľky: 'Grafová neurónová sieť'. Pre klasifikáciu uzlov lepšie fungovala základná varianta grafového konvolučného operátoru. Pre klasifikáciu grafov zase lepšie fungoval relačný grafový konvolučný operátor. Klasifikovanie spojených podgrafov som sa snažil obohatiť o predikovanie počtu malware uzlov v podgrafe (bol vytvorený takzvaný multi-task model, ktorý okrem binárnej klasifikácie riešil aj predikciu spojitej hodnoty). Dopadlo to, tak že sa sieť naučila predikovať iba samé nuly, pretože väčšina podgrafov nemá malware. To, že sa sieť trénovala s ďalším ground truth, vyjadrujúcim ako moc je podgraf zasiahnutý malwareom, však modelu pomohlo zlepšiť samotnú klasifikáciu (viď metriky FNR a AUC v poslednom riadku uvedenej tabuľky). Ďalšou zaujímavosťou zistenou pri trénovaní grafových neurónových sietí bolo to, že sú strašne citlivé na vysoký dropout. Pri nastavení dropoutu: $p = 0,5$ (nastavenie použité pri trénovaní dopredných neurónových sietí) sa sieť nedokázala naučiť skoro nič, preto musel byť dropout znížený: $p = 0,2$.

Prvok	Grafová neurónová sieť	Loss funkcia	Presnosť [%]	FNR [%]	AUC
uzol	2x Obyčajný grafový konvolučný operátor	Focal loss	99,281	**1,931**	**0,987**
	2x Relačný grafový konvolučný operátor	Focal loss	**99,456**	3,310	0,981
graf	2x Relačný grafový konvolučný operátor + Max pooling uzlových reprezentácií	Focal loss	95,573	12,052	0,929
podgraf	2x Relačný grafový konvolučný operátor + Max pooling uzlových reprezentácií	Focal loss	98,530	10,277	0,943
podgraf po spájaní	2x Relačný grafový konvolučný operátor + Max pooling uzlových reprezentácií	Focal loss	**98,884**	6,877	0,961
podgraf po spájaní	2x Relačný grafový konvolučný operátor + Mean pooling uzlových reprezentácií	Focal loss + MSE	97,964	**4,676**	**0,967**

Obr. 6.7: Výsledky klasifikácie rôznych typov grafových neurónových sietí, ktoré boli natrénované na charakteristikách a štruktúre uzlov v grafoch. V prípade klasifikácie grafov a podgrafov bolo nutné použiť pooling vrstvu pre vytvorenie grafovej vektorovej reprezentácie zo všetkých vektorových reprezentácií uzlov, ktoré vyšli z druhého grafového konvolučného operátoru. Všetky tieto modely mali spoločné to, že ich výstup z grafových konvolučných operátorov ešte smeroval do doprednej neurónovej siete s 2 skrytými vrtvami, ktorá následne klasifikovala prvok do jednej z tried.

Výsledky klasifikácie dopredných neurónových sietí, natrénovanými nad vektorovými reprezentáciami prvkov, ktoré vznikli metódami učenia bez učiteľa, sú zobrazené v tabuľke 6.8. **Predspracovanie metódami učenia bez učiteľa výrazne pomohlo pri klasifikácií uzlov**. Dopredná neurónová sieť natrénovaná nad vektormi z výstupu grafového autoenkodéra dosahovala najlepšie výsledky naprieč všetkými experimentami pre klasifikáciu uzlov (viď riadok uvedenej tabuľky pre *GAE* vektory: $AUC = 0.99$ a $FNR = 1.422$). **Pre klasifikáciu grafov dosahovali najlepšie výsledky grafové neurónové siete, predspracovanie metódami učenia bez učiteľa tu nepomohlo.** Veľmi dobre si obstáli aj *Node2Vec* vektory, čo sú vektory, v ktorých je zakódovaná iba štruktúra uzlov. Výsledky klasifikácie nad *Node2Vec* vektormi potvrdzujú, že aj v grafovej štruktúre sa nachádza dostatok informácií potrebných pre rozlíšenie medzi čistými a malware uzlami. Nevýhodou *Node2Vec* vektorov, je že vektory z viacerých grafov nemôžeme pre trénovanie kombinovať (vo vektoroch je, pri trénovaní modelu, vždy zakódovaná iba štruktúra jedného grafu, viď podkapitola 4.2).

Prvok	Predspracované vektorové reprezentácie	Presnosť [%]	FNR [%]	AUC
uzol	BFS Node2Vec vektory	93,8	10,8	0,915
	BFS Node2Vec vektory + vektory charakteristík	99,1	8,4	0,955
	GAE vektory	99,331	**1,422**	**0,99**
	VGAE vektory	**99,571**	21,223	0,893
podgraf po spájaní	priemer GAE vektorov	89,555	7,937	0,907

Obr. 6.8: Výsledky klasifikácie dopredných neurónových sietí s dvomi skrytými vrstvami, natrénovanými nad vektorovými reprezentáciami prvkov, ktoré vznikli pomocou metód bez učiteľa.

Kapitola 7

Záver

Cieľ práce, navrhnúť systém pre dolovanie znalostí z databázy behaviorálnych grafov, ktorý sa skladá iba z metód učenia bez učiteľa, bol splnený. Navrhnutý systém dokáže nad grafovou databázou vytvárať zhluky grafov, v ktorých sa malware a čisté prvky takmer vôbec nemiešajú. Systém pre dolovanie znalostí z grafov vyzerá nasledovne: Grafy sú na vstupe systému dekomponované na menšie podgrafy pomocou metódy *EdMot*. Ďalšou časťou systému je variačný grafový autoenkodér, ktorý z podgrafov vytvára vektorové reprezentácie. Poslednou časťou systému je zhlukovacia metóda *MeanShift*, ktorá z vektorových reprezentácií podgrafov vytvára zhluky. Pre výber metódy každej z častí systému boli vykonané experimenty. Metódy, ktoré boli vybraté do systému, dosahovali najlepších výsledkov u skúmaných metrík. Napríklad kombinácia metód *EdMot* a *Louivain* dosahovala, spomedzi všetkých testovaných metód, najlepšej hodnoty u metriky modularita (metrika hodnotiaca kvalitu nájdených komunít). Pre ohodnotenie kvality výstupov systému bola v práci vytvorená vlastná metrika nazvaná jednofarebnosť, ktorá hodnotí miešanie tried v každom zo zhlukov na výstupe. Variačný grafový autoenkodér bol ako časť systému vybraný pretože, po aplikovaní zhlukovacej metódy nad jeho vektorovými reprezentáciami podgrafov, systém dosahoval najlepších výsledkov u metriky jednofarebnosť.

Výsledky experimentov, ktoré mali zistiť, či metódy učenia bez učiteľa dokážu pomôcť modelom učenia s učiteľom, sú nasledujúce: Pre klasifikáciu uzlov, metódy bez učiteľa výrazne pomohli. Model, ktorý dosahoval najlepších výsledkov ($AUC = 0.99$) bol natrénovaný nad vektormi, ktoré vyšli z grafového autoenkodéra (metóda učenia bez učiteľa). Pre klasifikáciu grafov, metódy učenia bez učiteľa príliš nepomohli. Model, ktorý dosahoval najlepších výsledkov ($AUC = 0.967$) bola grafová neurónová sieť s dvomi grafovým relačným operátormi.

Čo sa týka možného pokračovania v práce, tak vhodným rozšírením by bolo nájsť náhradu za zhlukovaciu metódu *MeanShift*, ktorá ponúka množstvo výhod ako to, že je bezparametrická a experimentami prevedenými v práci sa potvrdilo, že v priestore dátových bodov dokáže nájsť rozumné zhluky. Jej nevýhoda je v tom, že pre zhlukovanie veľkého množstva dátových bodov je pomalá. Časová zložitosť metódy je $\mathcal{O}(n^2)$.

Literatúra

[1] *Simplified molecular-input line-entry system.* 2020. [Online; navštívené 5.12.2020]. Dostupné z: https://en.wikipedia.org/wiki/Simplified_molecular-input_line-entry_system.

[2] *Kullback–Leibler divergence.* 2021. [Online; navštívené 11.4.2021]. Dostupné z: https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence.

[3] Arora, A. *What is Focal Loss and when should you use it?* 2020. [Online; navštívené 10.4.2021]. Dostupné z: https://amaarora.github.io/2020/06/29/FocalLoss.html#where-was-focal-loss-introduced-and-what-was-it-used-for.

[4] Chen, W., Hu, X., Chen, W., Hong, Y. a Yang, M. *Airborne LiDAR Remote Sensing for Individual Tree Forest Inventory Using Trunk Detection-Aided Mean Shift Clustering Techniques.* Remote Sensing. *Júl 2018, zv. 10, s. 1078. DOI: 10.3390/rs10071078.*

[5] Elinas, *P. Knowing Your Neighbours: Machine Learning on Graphs. 2020. [Online; navštívené 5.12.2020]. Dostupné z: https://www.kdnuggets.com/2019/08/neighbours-machine-learning-graphs.html.*

[6] Emmons, *S.*, Kobourov, *S.*, Gallant, *M. a* Borner, *K. Analysis of Network Clustering Algorithms and Cluster Quality Metrics at Scale.* PLOS ONE. *Máj 2016, zv. 11. DOI: 10.1371/journal.pone.0159161.*

[7] Hamilton, *W. L.*, Ying, *R. a* Leskovec, *J. Representation Learning on Graphs: Methods and Applications. 2018.*

[8] Han, *F. Tutorial on Variational Graph Auto-Encoders. 2019. [Online; navštívené 11.4.2021]. Dostupné z: https://towardsdatascience.com/tutorial-on-variational-graph-auto-encoders-da9333281129.*

[9] Hodler, *M. N. . A. E. Graph Algorithms in Neo4j: Louvain Modularity. 2019. [Online; navštívené 10.3.2021]. Dostupné z: https://neo4j.com/blog/graph-algorithms-neo4j-louvain-modularity/.*

[10] Kipf, *T. N. a* Welling, *M. Variational Graph Auto-Encoders. 2016.*

[11] Kipf, *T. N. a* Welling, *M. Semi-Supervised Classification with Graph Convolutional Networks. 2017.*

[12] Li, *P.-Z.*, Huang, *L.*, Wang, *C.-D. a* Lai, *J.-H. EdMot.* Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.

ACM. Jul 2019. DOI: 10.1145/3292500.3330882. Dostupné z: http://dx.doi.org/10.1145/3292500.3330882.

[13] Navone, *E. C. Dijkstra's Shortest Path Algorithm. 2020. [Online; navštívené 1.5.2021]. Dostupné z: https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/.*

[14] Raghavan, *U. N.*, Albert, *R. a* Kumara, *S. Near linear time algorithm to detect community structures in large-scale networks.* Physical Review E. *American Physical Society (APS). Sep 2007, zv. 76, č. 3. DOI: 10.1103/physreve.76.036106. ISSN 1550-2376. Dostupné z: http://dx.doi.org/10.1103/PhysRevE.76.036106.*

[15] Schlichtkrull, *M.*, Kipf, *T. N.*, Bloem, *P.*, Berg, *R. van den*, Titov, *I. et al. Modeling Relational Data with Graph Convolutional Networks. 2017.*

[16] Tong, *F. Everything you need to know about Graph Theory for Deep Learning. 2019. [Online; navštívené 5.12.2020]. Dostupné z: https://towardsdatascience.com/graph-theory-and-deep-learning-know-hows-6556b0e9891b.*

[17] Tong, *F. What is Geometric Deep Learning? 2019. [Online; navštívené 5.12.2020]. Dostupné z: https://flawnsontong.medium.com/what-is-geometric-deep-learning-b2adb662d91d.*