UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ZHLUKOVANIE ČASOVÝCH RADOV ZALOŽENÉ NA ALGORITMOCH DETEKCIE KOMUNÍT V SOCIÁLNYCH SIEŤACH

DIPLOMOVÁ PRÁCA

2020 Bc. Anna Mária HOLIENČINOVÁ

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ZHLUKOVANIE ČASOVÝCH RADOV ZALOŽENÉ NA ALGORITMOCH DETEKCIE KOMUNÍT V SOCIÁLNYCH SIEŤACH

DIPLOMOVÁ PRÁCA

Študijný program:	Ekonomicko-finančná matematika a modelovanie
Študijný odbor:	1114 Aplikovaná matematika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Vedúci práce:	doc. RNDr. Beáta Stehlíková, PhD.

Bratislava 2020 **Bc. Anna Mária HOLIENČINOVÁ**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Anna Mária Holienčinová
Študijný program: ekonomicko-finančná matematika a modelovanie (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: aplikovaná matematika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Zhlukovanie časových radov založené na algoritmoch detekcie komunít v sociálnych sieťach
Time series clustering based on community detection algorithms in social networks

Anotácia: V práci sa vysvetľuje algoritmus zhlukovania časových radov navrhnutý v článku Ferreira a Zhao (2016) a implementuje sa použitím rôznych metód vytvorenia siete a detekovania komunít, ktoré boli v tomto článku uvažované. Ako vlastný prínos diplomovej práce sa skúma otázka, či sa nedá algoritmus vylepšiť tým, že namiesto jeho „najlepšej“ verzie sa ich zrealizuje niekoľko a vhodným spôsobom sa skombinujú. Súčasťou práce je aplikácia na testovacích dátach, ako aj na vlastných dátach týkajúcich sa zvolenej zaujímavej a aktuálnej problematiky.

Vedúci: doc. RNDr. Mgr. Beáta Stehlíková, PhD.
Katedra: FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry: prof. RNDr. Marek Fila, DrSc.

Dátum zadania: 08.01.2019

Dátum schválenia: 08.01.2019 prof. RNDr. Daniel Ševčovič, DrSc.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie Chcela by som sa poďakovať svojej vedúcej diplomovej práce doc. RNDr. Beáte Stehlíkovej, PhD. za trpezlivosť, cenné odborné rady a pripomienky a priateľský prístup, ktoré mi pomohli pri písaní tejto práce. Ďakujem aj mojim rodičom, manželovi a priateľom za ich trpezlivosť a podporu počas celého štúdia.

Abstrakt v štátnom jazyku

HOLIENČINOVÁ, Anna Mária: Zhlukovanie časových radov založené na algoritmoch detekcie komunít v sociálnych sieťach [Diplomová práca], Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky; školiteľ: doc. RNDr. Beáta Stehlíková, PhD., Bratislava, 2020, 75 s.

Cieľom tejto diplomovej práce je vysvetliť algoritmus zhlukovania časových radov navrhnutý v článku [8] a implementovať ho použitím rôznych metód vytvorenia siete a detekcie komunít. Zároveň skúmame otázku, či sa nedá algoritmus vylepšiť kombináciou rôznych realizácií algoritmu. Algoritmus aplikujeme na testovacích aj reálnych dátach, ktorými sú výnosy akcií indexu S&P 500 v roku 2018.

Diplomová práca má 4 kapitoly. V prvej kapitole uvádzame čitateľa do problematiky, vysvetľujeme teoretické poznatky potrebné k pochopeniu algoritmu. V druhej kapitole vysvetľujeme fungovanie algoritmu navrhnutého v článku [8] a navrhujeme vylepšenie a zostrojenie klasifikátora. V tretej kapitole aplikujeme algoritmus na testovacie dáta a podrobne vysvetľujeme vylepšenie algoritmu. V štvrtej kapitole aplikujeme algoritmus na reálne dáta a hodnotíme získané výsledky.

Kľúčové slová: zhlukovanie, časové rady, komunity, sociálne siete

Abstract

HOLIENČINOVÁ, Anna Mária: Time series clustering based on community detection algorithms in social networks [Master Thesis], Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Mathematics and Statistics; Supervisor: doc. RNDr. Beáta Stehlíková, PhD., Bratislava, 2020, 75 p.

The aim of this master thesis is to explain the time series clustering algorithm proposed in [8] and implement it using various methods of creating networks and community detection algorithms. Also, we are investigating whether the algorithm can be improved by combining different algorithm implementations. We apply the algorithm to both test and real data, which are the S&P 500 stock returns in 2018.

The master thesis has 4 chapters. In the first chapter we introduce the theoretical knowledge needed to understand the algorithm. In the second chapter, we explain the algorithm proposed [8] and propose improvements and construction of the classifier. In the third chapter we apply the algorithm to the test data and explain the improvement of the algorithm. In the fourth chapter we apply the algorithm to real data and evaluate the obtained results.

Keywords: clustering, time series, communities, social networks

Obsah

Úvod **8**

1 Teoretický úvod do problematiky **10**

1.1 Úvod do analýzy zhlukov časových radov 10

1.2 Miery vzdialenosti 10

1.2.1 Shape-based miery vzdialenosti 11

1.2.2 Edit-based miery vzdialenosti 14

1.2.3 Feature-based miery vzdialenosti 15

1.2.4 Structure-based miery vzdialenosti 16

1.3 Vytvorenie siete 16

1.4 Detekcia komunít v sieťach 17

2 Algoritmus zhlukovania časových radov a klasifikácia **23**

2.1 Výpočet zhôd algoritmov 24

2.2 Vážená sieť na základe zhôd algoritmov 24

2.3 Zostrojenie klasifikátora 24

3 Aplikácia algoritmu na testovacie dáta **25**

3.1 Dataset *coffee* 25

3.2 Dataset *gunpoint* 30

3.3 Ďalšie datasety 38

4 Aplikácia algoritmu na reálne dáta **43**

Záver **52**

Zoznam použitej literatúry **53**

Príloha **56**

Úvod

Dáta vo forme časových radov dnes nájdeme takmer v každom odvetví a odvetvie financií nie je výnimkou. Ceny a výnosy akcií, úrokové miery, ale napríklad aj dáta o počte predaných produktov - to všetko sú časové rady. Časové rady môžeme analyzovať rôznymi spôsobmi. Jednou zo základných metód dátovej analýzy časových radov je aj zhlukovanie.

V tejto diplomovej práci sa venujeme jednému z alternatívnych prístupov k zhlukovaniu časových radov a to pomocou algoritmov detekcie komunít v sociálnych sieťach. Týmto prístupom k zhlukovaniu sa zaoberali autori článku [8], v ktorom ukázali, že tento prístup predstavuje efektívny spôsob, ktorý môže v určitých prípadoch dosiahnuť lepšie výsledky ako klasické metódy zhlukovania. V diplomovej práci sa snažíme túto metódu vylepšiť, a to rozšírením algoritmu o výpočet zhôd algoritmov. Na datasety aplikujeme algoritmus navrhnutý v článku [8] niekoľkokrát s rôznym nastavením parametrov vstupujúcich do algoritmu, vypočítame koľkokrát sa každé dva časové rady dostali do spoločného zhluku a na základe toho vytvoríme novú sociálnu sieť. Hrany v tejto sociálnej sieti vzniknú na základe hranice existencie, ktorá predstavuje koľkokrát sa algoritmy pri zhlukovaní zhodli a zaradili časové rady do spoločných zhlukov.

Algoritmus spolu s klasifikátorom sme aplikovali na vybrané datasety z z UCR Time Series Classification Archive [12]. Tieto dáta, narozdiel od reálnych dát, obsahujú informáciu o skutočnom rozdelení dát do zhlukov, preto sú vhodné na testovanie úspešnosti vylepšeného algoritmu a klasifikátora. Okrem toho aplikujeme algoritmus zhlukovania aj na reálne dáta - denné výnosy cien akcií indexu S&P 500 za rok 2018.

V prvej kapitole vysvetľujeme základné pojmy zo zhlukovania časových radov, ktoré sú potrebné na lepšie pochopenie algoritmu a prezentujeme prehľad mier vzdialenosti a algoritmov detekcie komunít, ktorými sa v práci bližšie zaoberáme.

V druhej kapitole predstavujeme algoritmus navrhnutý v článku [8] a proces, ktorým zisťujeme zhodu algoritmov. Súčasťou tejto kapitoly je aj popis konštrukcie jednoduchého klasifikátora, pomocou ktorého klasifikujeme testovacie dáta na základe toho, ako algoritmus klasifikoval trénovacie dáta.

V tretej kapitole uvádzame podrobnú ilustráciu algoritmu na vybrané datasety *coffee* a *gunpoint* z [12], ako aj zhrnutie výsledkov na ďalšie datasety z [12].

Výsledky, ktoré sme dostali aplikovaním algoritmu na reálne dáta, ktorými sú výnosy akcií indexu S&P 500, uvádzame v štvrtej kapitole tejto diplomovej práce. Súčasťou práce je aj skript v jazyku R.

1 Teoretický úvod do problematiky

1.1 Úvod do analýzy zhlukov časových radov

Analýza zhlukov je jednou zo základných metód dátovej analýzy, ktorá sa zaoberá hľadaním podobnosti objektov s väčším množstvom premenných. Medzi takéto viacrozmerné objekty zaraďujeme aj časové rady. Pomocou zhlukovej analýzy vieme nájsť v dátach určité súvislosti, na základe ktorých môžeme dáta klasifikovať do zhlukov (klastrov). Dáta, klasifikované do rovnakého zhluku by mali byť medzi sebou čo najviac podobné a zároveň by mali byť čo najviac rozdielne od dát klasifikovaných do iných zhlukov na základe určenej miery podobnosti.

Nech n je počet objektov a k počet zhlukov. Predpokladáme, že $n \geq k$, teda že počet objektov je väčší ako počet zhlukov. Zhlukovanie definujeme nasledovne [11]:

Definícia 1.1. *Nech $\Gamma_{n,k}$ je množina všetkých takých vektorov $\gamma \in \mathbb{R}^n$ s prvkami z množiny $\{1, \ldots, k\}$, že každá hodnota $1, \ldots, k$ sa vo vektore γ nachádza aspoň raz. Vektor $\gamma \in \Gamma_{n,k}$ budeme nazývať zhlukovaním. Množinu $C_j(\gamma) = \{i \in \{1, \ldots, n\} : \gamma_i = j\}$ budeme nazývať j-ty zhluk zhlukovania γ. Počet $n_j(\gamma)$ prvkov $C_j(\gamma)$ budeme nazývať veľkosť j-teho zhluku zhlukovania γ.*

Dáta vo forme časového radu definujeme nasledovne [8]:

Definícia 1.2. *Časový rad X je usporiadaná postupnosť N reálnych hodnôt $X = \{x_1, x_2, \ldots, x_N\}$, $x_i \in \mathbb{R}$, $i \in \mathbb{N}$.*

Pri zhlukovaní časových radov sú objektami zhlukovania práve časové rady. V spoločnom zhluku by mali byť časové rady, ktoré sú medzi sebou podobné. Na meranie tejto podobnosti medzi časovými radmi potrebujeme nejakú mieru blízkosti alebo vzdialenosti, ktoré bližšie popisujeme v 1.2.

1.2 Miery vzdialenosti

Mieru vzdialenosti, resp. podobnosti medzi časovými radmi meriame pomocou určitých vzdialeností medzi časovými radmi. Miery vzdialenosti rozdeľujeme do štyroch kategórií [8] na základe toho, akým spôsobom merajú túto vzdialenosť v časových radoch.

1.2.1 Shape-based miery vzdialenosti

Shape-based miery vzdialenosti [8] medzi časovými radmi merajú vzdialenosť na základe tvaru časových radov. Medzi tieto miery vzdialenosti patrí aj L_p-norma, ktorá je definovaná nasledovne:

$$d(X,Y) = \left(\sum_{i=1}^{N} |x_i - y_i|^p \right)^{\frac{1}{p}}, \tag{1.1}$$

kde p celé kladné číslo, $X = (x_1, x_2, \ldots, x_N)$ a $Y = (y_1, y_2, \ldots, y_N)$ sú časové rady. Tieto miery vzdialenosti merajú vzdialenosť medzi dvomi fixnými bodmi časového radu, čo je zároveň aj ich najväčšou nevýhodou. Dva časové rady s rovnakým priebehom líšiace sa posunom na osi y vyhodnotia tieto miery vzdialenosti ako veľmi odlišné. Z tohto dôvodu sa nazývajú aj *lock-step* mierami vzdialenosti.

Medzi najpoužívanejšie L_p-normy patria

- Manhattanská (poštárska) norma pre $p = 1$

$$d(X,Y) = \sum_{i=1}^{N} |x_i - y_i|, \tag{1.2}$$

- Euklidovská norma pre $p = 2$

$$d(X,Y) = \sqrt{\sum_{i=1}^{N} |x_i - y_i|^2}, \tag{1.3}$$

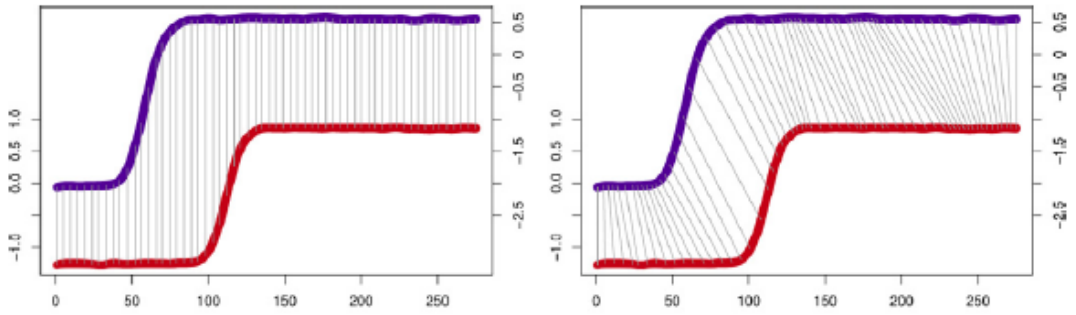
- maximová norma pre $p \to \infty$

$$d(X,Y) = \max_{i \in \{1,\ldots,N\}} |x_i - y_i| . \tag{1.4}$$

Robustnejšie výsledky porovnania poskytujú *elastické* miery vzdialenosti, medzi ktoré patrí Dynamic Time Warping (DTW) [8].

Dynamic Time Warping je algoritmus, pomocou ktorého môžeme merať vzdialenosť medzi časovými radmi, pri ktorých je prítomný fázový posun. Vieme teda nájsť podobnosť aj v takých časových radoch, kde jeden z nich obsahuje zrýchlenia alebo spomalenia. Táto miera vzdialenosti časové rady zarovná a zároveň vypočíta vzdialenosť medzi nimi.

Rozdiel medzi *lock-step* mierami vzdialenosti a algoritmom DTW je znázornený na Obr.1.1.



Obr. 1.1: Ilustrácia rozdielu medzi *lock-step* mierami vzdialenosti (vľavo) a algoritmom Dynamic Time Warping (vpravo), zdroj: [8]

Majme dva časové rady, $X = (x_1, x_2, \ldots, x_N)$ a $Y = (y_1, y_2, \ldots, y_M)$. Časový rad X umiestnime na os x, Y umiestnime na os y a oba časové rady rozdelíme po bodoch, vďaka čomu nám vznikne mriežka. Cieľom algoritmu je nájsť takú „warping cestu“ z bodu (1, 1) do bodu (N, M), ktorá bude minimalizovať celkovú vzdialenosť medzi danými časovými radmi. DTW algoritmus teda môžeme zapísať takto [14]:

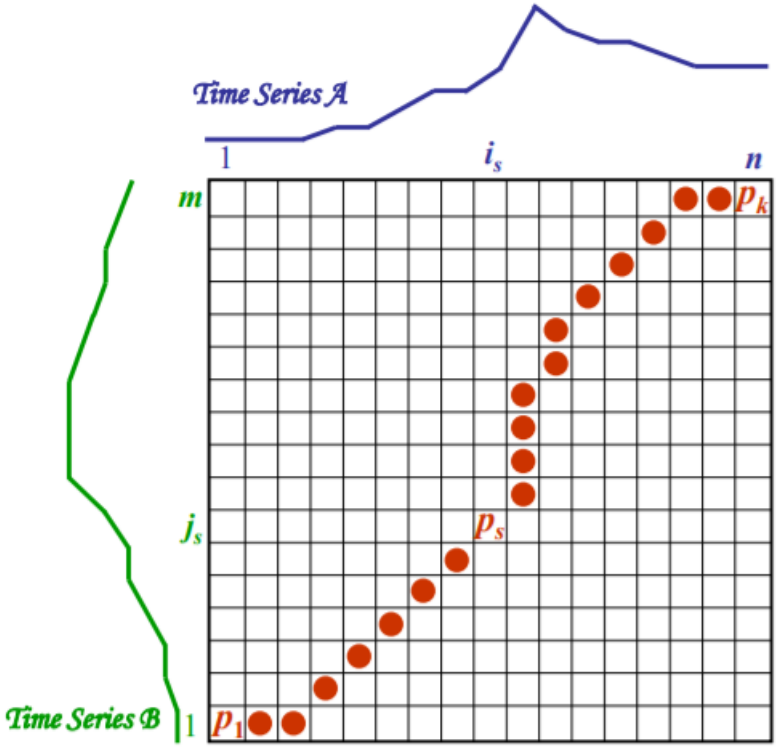
1. vypočítame (euklidovskú) vzdialenosť medzi prvým bodom prvého časového radu a všetkými bodmi druhého časového radu a vyberieme minimálnu vzdialenosť,
2. zoberieme ďalší bod prvého časového radu a budeme opakovať krok 2,
3. opakujeme kroky 2 a 3 dovtedy, kým použijeme všetky body z prvého časového radu,
4. budeme opakovať kroky 2 a 3, ale referenčným časovým radom bude druhý časový rad,
5. sčítame minimálne vzdialenosti a dostávame mieru vzdialenosti medzi časovými radmi.

Počet takýchto „warping ciest“ narastá exponenciálne a preto je potrebné brať do úvahy určité obmedzenia [3]:

- **monotónnosť**: body musia byť usporiadané podľa časovej zložky, teda $x_{i-1} \leq x_i$ a zároveň $y_{i-1} \leq y_i$,
- **spojitosť**: $x_i - x_{i-1} \leq 1$ a zároveň $y_i - y_{i-1} \leq 1$,

- **ohraničenia**: $x_1 = 1, x_t = N$ a zároveň $y_1 = 1, y_t = M$,
- **„warping okno“**: prípustné body môžu byť obmedzené tým, že musia spadať pod „warping okno“ $|x_i - y_i| \leq r$, kde $r > 0$ je dĺžka tohto okna,
- **obmedzenie sklonu**: prípustná „warping cesta“ môže byť obmedzená sklonom, aby sa zabránilo príliš veľkým pohybom v jednom smere

Ilustrácia hľadania optimálnej „warping cesty“ je zobrazená na Obr.1.2.



Obr. 1.2: Ilustrácia hľadania optimálnej „warping cesty“ pri algoritme DTW, zdroj: [28]

Medzi ďalšie *elastické* shape-based miery vzdialenosti patria:

- Short Time Series (STS) [19] je miera definovaná ako

$$d(X,Y) = \sqrt{\sum_{i=0}^{N-1} \left(\frac{y_{i+1} - y_i}{t_{i+1} - t_i} - \frac{x_{i+1} - x_i}{t_{i+1} - t_i} \right)^2}, \tag{1.5}$$

  kde $(t_{i+1} - t_i)$ je rozdiel medzi dvomi časovými bodmi. Táto miera vzdialenosti je preto vhodná aj pre také dva časové rady, ktoré majú rovnaký počet časových bodov - teda sú rovnako dlhé, ale časové body nie sú rovnomerne distribuované na časovej osi.

- DISSIM [10] vypočítame ako určitý integrál z Euklidovskej vzdialenosti podľa času, teda

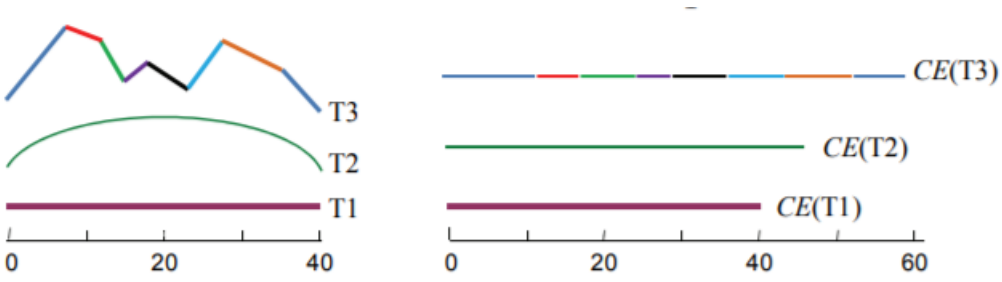
$$d(X,Y) = \int_{t_1}^{t_N} D_{X,Y}(t)dt, \tag{1.6}$$

kde $D_{X,Y}$ je definované ako Euklidovská norma, pričom predpokladáme, že medzi jednotlivými časovými bodmi sú časové rady lineárne. Pri výpočte vzdialenosti pomocou tejto miery vzdialenosti je potrebné, aby časové rady začínali a končili v rovnakom časovom intervale, ale nemusia byť rozdelené na rovnaké časové body a ich časové body nemusia byť distribuované rovnomerne na časovej osi.

- Complexity Invariant Distance (CID) [2] je norma založená na Euklidovskej vzdialenosti upravenej o odhad zložitosti časového radu a vypočítame ju ako

$$d(X,Y) = D(X,Y) \cdot CF(X,Y), \tag{1.7}$$

kde $D(X,Y)$ je Euklidovská vzdialenosť, $CF(X,Y) = \frac{\max\{(CE(X),CE(Y)\}}{\min\{(CE(X),CE(Y)\}}$ a $CE(Q)$ je odhad zložitosti časového radu $Q = (q_1, q_2, \ldots, q_N)$, ktorý vypočítame ako $CE(Q) = \sqrt{\sum_{i=1}^{N-1}(q_i - q_{i+1})^2}$. Odhad zložitosti časového radu je založený na úvahe, že ak by sme mohli „naťahovať“ časový rad až do momentu, pokiaľ sa z neho stane rovná čiara, zložitejší časový rad by mal túto čiaru dlhšiu ako jednoduchší časový rad. Ilustrácia odhadu zložitosti časového radu je znázornená na Obr. 1.3.



Obr. 1.3: Ilustrácia odhadu zložitosti časového radu pri algoritme CID, zdroj: [2]

1.2.2 Edit-based miery vzdialenosti

Edit-based miery vzdialenosti [8] počítajú vzdialenosť medzi časovými radmi tak, že vypočítajú minimálny počet operácií, ktoré je nutné urobiť, aby sme jeden časový

rad transformovali na druhý. Medzi takéto operácie zaraďujeme vkladanie, odstránenie alebo nahradenie postupnosti v časovom rade. Týmito mierami vzdialenosti sa ale nebudeme v našej práci bližšie zaoberať.

1.2.3 Feature-based miery vzdialenosti

Pri zhlukovaní časových radov sa často stretávame s problémom, že vo viacrozmernom priestore sa strácajú rozdiely medzi časovými radmi. Navyše, mnohorozmernosť spôsobuje, že zhlukovacie algoritmy sú málo efektívne. Feature-based miery vzdialenosti [8] počítajú vzdialenosť tak, že extrahujú z časových radov určité „črty“ vďaka čomu znižujú dimenziu dát a následne merajú vzdialenosť medzi týmito črtami. Medzi tieto miery vzdialenosti patria:

- Discrete Wawelet Transform (DWT) [30] - pri porovnávaní vzdialenosti časových radov pomocou DWT nahradíme časový rad jeho vlnovou transformáciou a následne porovnávame vzdialenosť medzi nimi pomocou Euklidovskej vzdialenosti. Vlnové transformácie, narozdiel od Fourierových transformácií rozkladajú časový rad nielen na základe jeho frekvencie, ale aj na základe jeho časovej zložky. To znamená, že zatiaľčo pri Fourierovej transformácii časového radu dostávame informáciu len o tom, aké frekvencie sú v časovom rade prítomné, pri vlnovej transformácii dostávame informáciu aj o tom, v ktorom časovom úseku sa tieto frekvencie nachádzajú [27].

- Korelácia [16] - na porovnávanie dát pomocou korelácie budeme používať jej transformáciu:
$$d(X,Y) = \sqrt{2 - 2\mathrm{cor}(X,Y)}. \tag{1.8}$$

- Integrated periodogram (INTPER) [17] - algoritmus, ktorý počíta vzdialenosť medzi dvomi rovnako dlhými časovými radmi ako vzdialenosť medzi ich integrovanými periodogramami. Periodogram sa používa na identifikovanie tzv. dominantných periód časových radov [25]. Vzdialenosť je počítaná ako
$$d(X,Y) = \int_{-\pi}^{\pi} |F_X(\lambda) - F_Y(\lambda)|\, d\lambda, \tag{1.9}$$
kde $F_X(\lambda_j) = \frac{\sum_{i=1}^{j} I_X(\lambda_i)}{C_X}$, $F_Y(\lambda_j) = \frac{\sum_{i=1}^{j} I_Y(\lambda_i)}{C_Y}$ a $C_X = \sum_i I_X(\lambda_i)$, $C_Y = \sum_i I_Y(\lambda_i)$

pre normalizovanú verziu algoritmu a $C_X = C_Y = 1$ pre nenormalizovanú verziu algoritmu. I_X a I_Y označujú periodogramy časových radov X a Y, ktoré sú definované ako

$$I_X(\lambda_k) = \frac{\left|\sum_{t=1}^{N} X_t \mathrm{e}^{-i\lambda_k t}\right|^2}{N} \tag{1.10}$$

$$I_Y(\lambda_k) = \frac{\left|\sum_{t=1}^{N} Y_t \mathrm{e}^{-i\lambda_k t}\right|^2}{N}, \tag{1.11}$$

kde $\lambda_k = \frac{2\pi k}{N}$, $k = 1, \ldots, n$ pre $n = \frac{N-1}{2}$. Odporúča sa používať normalizovanú verziu vtedy, keď sa grafy časových radov pretínajú a nenormalizovanú verziu vtedy, keď sa nepretínajú [7].

1.2.4 Structure-based miery vzdialenosti

Structure-based miery vzdialenosti [8] sú založené na parametrických modeloch, napríklad Hidden Markov Models alebo ARMA. Týmito meriami sa ale podobne ako s edit-based mierami nebudeme v našej práci bližšie zaoberať.

1.3 Vytvorenie siete

Časové rady v našej práci zobrazujeme pomocou grafov, resp. sietí. Sieť predstavuje užitočný mechanizmus na reprezentáciu dát ako vrcholov a vzťahov medzi nimi, ktoré sú reprezentované prostredníctvom hrán. V našom prípade sú vrcholmi siete jednotlivé časové rady a vzťahy medzi časovými radmi predstavujú vzdialenosti, ktoré sme definovali vyššie.

Sieť je podľa [8] definovaná nasledovne:

Definícia 1.3. *Sieť (resp. graf) G(V,E) je zložená z množiny n vrcholov* $V = \{v_1, v_2, \ldots, v_n\}$ *a z množiny m hrán* $E = \{(v_i, v_j) | v_i, v_j \in V\}$*, kde* (v_i, v_j) *je hrana medzi vrcholmi* v_i *a* v_j*.*

Sieť vytvárame dvomi spôsobmi - pomocou algoritmu k-nn a ε-nn.

- algoritmus k-nn - vrchol spojíme s k najbližšími, resp. najpodobnejšími susedmi, teda každý časový rad spojíme s k časovými radmi, ktoré sú k nemu najbližšie, resp. najpodobnejšie na základe vzdialenosti, resp. podobnosti vypočítanej podľa algoritmov popísaných vyššie,

- algoritmus ε-nn - dva vrcholy spojíme, pokiaľ je ich vzájomná podobnosť väčšia ako zvolená hraničná hodnota $\varepsilon \in (0, 1)$. V prípade, že pracujeme s maticou vzdialeností, dva vrcholy spojíme, pokiaľ je ich vzájomná vzdialenosť nižšia ako zvolená hraničná hodnota $\varepsilon \in (0, 1)$.

1.4 Detekcia komunít v sieťach

Sociálne siete ako grafy sú vo všeobecnosti globálne riedke, avšak obsahujú podgrafy, ktoré sú zvyčajne husté. To znamená, že vrcholy vrámci jedného podgrafu sú medzi sebou pospájané hustejšie a zároveň vrcholy medzi jednotlivými podgrafmi sú pospájané redšie. Tieto podgrafy nazývame komunitami.

Kvalitu štruktúry komunít, ktorú vypočítame, posudzujeme pomocou modularity, ktorú označujeme Q. Modularita Q [6] je definovaná ako:

$$Q = \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w), \tag{1.12}$$

kde $k_v = \sum_w A_{vw}$ je stupeň vrchola v, $\delta(i, j)$ je tzv. Kroneckerova delta [13], ktorá nadobúda hodnotu 1 ak $i = j$ a 0 inak, $m = \frac{1}{2} \sum_{vw} A_{vw}$ je počet hrán v grafe a $A_{vw} = 1$, ak sú vrcholy v a w spojené a 0 inak, je prvok matice susednosti. „Najlepšie“ delenie vrcholov do komunít maximalizuje modularitu Q, teda sieť obsahuje veľa hrán vrámci komunity a málo hrán medzi komunitami.

Na detekciu komunít v sieťach používame nasledujúce algoritmy:

1. **Walktrap** [22] je algoritmus založený na náhodných prechádzkach po grafoch. Pod náhodnou prechádzkou po grafe rozumieme proces, pri ktorom sa chodec v každom kroku nachádza v nejakom vrchole a rovnomerne náhodne si vyberie z ostatných vrcholov vrchol, do ktorého sa presunie. Postupnosť navštívených vrcholov je Markovov reťazec, ktorého stavmi sú navštívené vrcholy grafu. Predpokladáme, že chodec počas náhodnej prechádzky po grafe má tendenciu zotrvať v hustejších častiach grafu, teda v komunite.

   Majme neorientovaný graf $G = (V, E)$, kde V je jeho množina vrcholov, pričom $n = |V|$, a E je množina jeho hrán, $m = |E|$. Graf G rozdelíme do n komunít, pričom každá z nich obsahuje práve jeden vrchol. Začíname z delenia

$\mathcal{P}_1 = \{\{v\}, v \in V\}$ a vypočítame vzdialenosť medzi priľahlými vrcholmi. Následne toto delenie rozvíjame v každom kroku k nasledovne:

- vyberieme dve komunity C_1 a C_2 z delenia $\mathcal{P}_k$ podľa kritéria založeného na vzdialenosti medzi komunitami,
- spojíme tieto dve komunity do novej komunity $C_3 = C_1 \cup C_2$ a vytvoríme nové delenie $\mathcal{P}_{k+1} = (\mathcal{P}_k \backslash \{C_1, C_2\}) \cup \{C_3\}$,
- znova vypočítame vzdialenosť medzi komunitami.

Po $n - 1$ krokoch dostaneme delenie $\mathcal{P}_n = \{V\}$. V každom kroku definujeme delenie grafu $\mathcal{P}_k$ na komunity, vďaka ktorému dostaneme hierarchickú štruktúru komunít nazývanú dendrogram. Dendrogram je stromový diagram, ktorého vetvy sú jednotlivé vrcholy grafu. Tie vetvy, ktoré sú navzájom prepojené, patria do spoločnej komunity.

Kľúčovým problémom algoritmu je určenie dvoch komunít, ktoré spojíme. Výber týchto komunít ovplyvňuje aj kvalitu štruktúry komunít, ktorú získame. Pre zjednodušenie, budeme spájať len susediace komunity, teda také, ktoré majú medzi sebou aspoň jednu hranu a komunity budeme vyberať pomocou Wardovho algoritmu.

2. **Fast&greedy** [6] algoritmus je rýchlejší a efektívnejší ako *walktrap* a podobne ako aj niektoré iné algoritmy založený na optimalizácii modularity. Zhlukovanie získané pomocou tohoto algoritmu je hierarchické.

   Definujeme podiel hrán spájajúcich vrcholy z komunity i s vrcholmi v komunite j ako

   $$e_{ij} = \frac{1}{2m} \sum_{vw} A_{vw} \delta(c_v, i) \delta(c_w, j) \tag{1.13}$$

   a podiel hrán, ktoré končia v komunite j ako

   $$a_i = \frac{1}{2m} \sum_{v} k_v \delta(c_v, i). \tag{1.14}$$

   Potom použitím $\delta(c_v, c_w) = \sum_i \delta(c_v, i)\delta(c_w, i)$ prepíšeme rovnicu (1.12) a dostá-

vame

$$Q = \frac{1}{2m} \sum v_i v_j \left[A_{v_i v_j} - \frac{k_{vi} k_{vj}}{2m} \sum_i \delta(c_v, i) \delta(c_w, i) \right] =$$
$$\sum_i \left[\frac{1}{2m} \sum_{vw} A_{vw} \delta(c_v, i) \delta(c_w, i) - \frac{1}{2m} \sum_v k_v \delta(c_v, i) \frac{1}{2m} \sum_w k_w \delta(c_w, i) \right]$$
$$= \sum_i \left(e_{ii} - a_i^2 \right) \quad (1.15)$$

Cieľom algoritmu je nájsť zmeny v modularite Q, ktorú by spôsobilo zlúčenie dvoch komunít, následne vyberieme to zlúčenie, ktoré spôsobí najvyššiu zmenu modularity a komunity zlúčime.

Uvažujme maticu ΔQ_{ij}, ktorá obsahuje zmeny modularity pre každú dvojicu i, j komunít, ktoré sú spojené aspoň jednou hranou. Táto matica je riedka čo značne šetrí výpočtový čas a pamäť. Ďalej uvažujme tzv. *max-heap* H, ktorý bude obsahovať najväčší prvok každého riadku ΔQ_{ij} spolu s označením i, j príslušnej dvojice komunít a vektor prvkov a_i.

V prvom kroku predpokladáme, že každý vrchol patrí do samostatnej komunity, teda $e_{ij} = \frac{1}{2m}$ ak i a j sú spojené a 0 inak a $a_i = \frac{k_i}{2m}$. Potom v prvom kroku algoritmu máme

$$\Delta Q_{ij} = \begin{cases} \frac{1}{2m} - \frac{k_i k_j}{(2m)^2} & \text{ak } i, j \text{ sú spojené,} \\ 0 & \text{inak.} \end{cases} \quad (1.16)$$

Na začiatku vypočítame počiatočné hodnoty ΔQ_{ij} a a_i a naplníme hodnotami *max-heap* H. Následne vyberieme najväčšie ΔQ_{ij} z H, zlúčime príslušné komunity, aktualizujeme maticu ΔQ, heap H a a_i a navýšime Q o ΔQ_{ij}. Tento krok opakujeme až pokiaľ nie sú všetky vrcholy v spoločnej komunite.

3. **Infomap**, algoritmus popísaný v článku [24], využíva na odhalenie komunít v sieti mapy. Majme váženú, orientovanú sieť, kde hrany reprezentujú interakciu a prenos informácie medzi jednotlivými vrcholmi. Skupinu vrcholov, medzi ktorými prúdi veľa informácií - teda je medzi nimi veľa hrán, budeme považovať za komunitu.

   Dôležitou súčasťou tohoto algoritmu je náhodný chodec, pričom pravdepodobnosť, že prejde z vrcholu v do vrcholu w je daná Markovovou maticou prechodu.

Vieme, že reálne siete obsahujú tzv. regióny, resp. podgrafy - teda časti grafu, do ktorých keď náhodný chodec vojde, má tendenciu v nich zostať a prechody medzi regiónmi nie sú časté. Cieľom je nájsť optimálne rozdelenie vrcholov do regiónov, tak aby sme nemali ani príliš veľa regiónov a ani príliš veľa vrcholov v jednotlivých regiónoch.

Hlavnou myšlienkou tohoto algoritmu je skomprimovanie informácií o dynamickom procese prebiehajúcom v grafe, ktorú nazývame náhodná prechádzka. To dosiahneme minimalizáciou funkcie nazvanej „Minimum Description Length“ náhodnej prechádzky alebo minimalizáciou „Map Equation“.

Infomap je algoritmus, ktorý je vhodný pre veľké siete do 100 000 vrcholov.

4. **Multilevel** [4] spomedzi všetkých ostatných algoritmov vyniká najmä vo výpočtovom čase.

   Uvažujme váženú sieť s n vrcholmi. V prvom kroku vytvoríme n komunít, v každej z nich bude práve jeden vrchol grafu. Potom pre každý vrchol v a susediaci vrchol w vypočítame vplyv na modularitu, ktorú spôsobí presun vrcholu v do komunity s vrcholom w. Vrchol v potom presunieme do takej komunity, pri ktorej bude tento vplyv maximálny. Pokiaľ je vplyv na modularity negatívny, vrchol v zostane v pôvodnej komunite. Tento proces opakujeme pre všetky vrcholy. Pokiaľ nám presun vrcholov do komunít už neprináša žiadne ďalšie zlepšenie modularity, prvý krok algoritmu je dokončený.

   Druhý krok algoritmu spočíva vo vytvorení novej siete, ktorej vrcholy budú komunity, ktoré sme našli počas prvého kroku. Váhy hrán medzi dvomi novými vrcholmi sú dané ako súčet váh hrán vrcholov pôvodnej siete, ktoré patria do jednotlivých komunít. Následne znova pokračujeme prvým krokom.

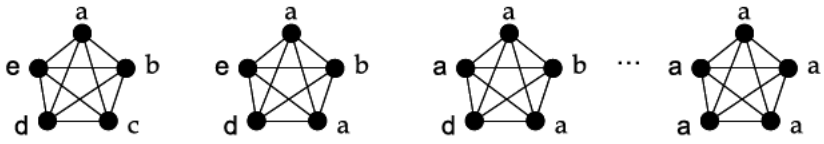
5. **Label propagation** je algoritmus založený na štítkoch (tzv. *labeloch*) jednotlivých vrcholov a spomína sa v článku [29]. Predpokladajme, že vrchol v má susediace vrcholy $v_1, v_2, \dots v_k$ a každý zo susediacich vrcholov má štítok, ktorý označuje do akej komunity vrchol patrí. Vrchol v si potom vyberie komunitu na základe toho, do ktorých komunít patria susediace vrcholy, pričom predpokladáme, že vrchol si vyberie tú komunitu, do ktorej patrí najviac z jeho susedov. V

prípade, že do dvoch komunít patrí rovnaký počet susediacich vrcholov, vrchol v si svoju komunitu vyberie rovnomerne náhodne.

V prvom kroku má každý vrchol rozdielny štítok, pričom jednotlivé vrcholy navzájom poznajú svoje štítky. V druhom kroku si vďaka tomu vrcholy, ktoré sú medzi sebou husto pospájané, vyberú jeden štítok, na ktorom sa zhodnú a zmenia si ho. Vrcholy, ktoré majú rovnaký štítok považujeme za komunitu.

Tento proces budeme iterovať, v každej iterácii si vrchol volí svoj štítok na základe štítkov susediacich vrcholov. Proces opakujeme dovtedy, pokiaľ má každý z vrcholov taký štítok, ktorý má väčšina z jeho susediacich vrcholov.

Ilustrácia algoritmu je zobrazená na Obr. 1.4.



Obr. 1.4: Ilustrácia algoritmu Label propagation, zdroj: [29]

6. **Optimal** [5] je algoritmom, ktorý je kvôli svojej exponenciálnej časovej zložitosti vhodný pre menšie siete - približne do 50 vrcholov. Hlavnou myšlienkou tohoto algoritmu je optimalizácia modularity. Tento problém vieme sformulovať ako úlohu celočíselného lineárneho programovania.

   Majme neorientovaný graf $G = (V, E)$, kde V je jeho množina vrcholov, pričom $n = |V|$, a E je množina jeho hrán, $m = |E|$. Definujeme „rozhodovacie“ premenné $X_{vw} \in \{0, 1\}$ pre všetky $v, w \in V$. Premenná $X_{vw} = 1$ práve vtedy, keď sú vrcholy v a w v tej istej komunite, inak nadobúda hodnotu 0. Chceme

maximalizovať modularitu Q definovanú ako (1.12) pri týchto ohraničeniach:

$$\text{reflexivita: } \forall\, v : X_{vv} = 1, \tag{1.17}$$

$$\text{symetria: } \forall\, v, w : X_{vw} = X_{wv}, \tag{1.18}$$

$$\text{tranzitivita: } \forall\, u, v, w : \begin{cases} X_{uv} + X_{vw} - 2X_{uw} \leq 1 \\ X_{uw} + X_{uv} - 2X_{vw} \leq 1 \\ X_{vw} + X_{uw} - 2X_{uv} \leq 1. \end{cases} \tag{1.19}$$

Ohraničenia, pri ktorých modularitu maximalizujeme, môžeme slovne popísať nasledovne [26]:

- reflexivita: vrchol je sám so sebou v rovnakej komunite,
- symetria: ak sú v rovnakej komunite vrcholy v a w, tak v rovnakej komunite sú aj w a v,
- tranzitivita: ak sú v rovnakej komunite vrcholy u a v a zároveň aj vrcholy v a w sú v rovnakej komunite, potom aj vrcholy u a w sú v rovnakej komunite. Keďže premenná X nadobúda iba hodnoty 0 a 1, zápis v tvare nerovnosti $X_{uv}+X_{vw}-2X_{uw} \leq 1$ môže byť porušený iba v prípade, že prvé dva sčítance sú rovné 1 (teda u, v sú v rovnakej komunite a v, w sú v rovnakej komunite) a odpočítavame 0 (teda u, w nie sú v rovnakej komunite) [26].

Účelovú funkciu, ktorú chceme maximalizovať vieme pomocou premenných X_{vw} prepísať do nasledovného tvaru:

$$Q = \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] X_{vw}. \tag{1.20}$$

Táto optimalizačná úloha sa pomocou funkcie `cluster_optimal` [20] v softvéri R rieši použitím GLPK knižnice.

2 Algoritmus zhlukovania časových radov a klasifikácia

Algoritmus z článku [8] je navrhnutý nasledovne:

1. **Normalizácia a úprava dát** - predstavuje pripravenie dát do podoby, aby sme s nimi boli schopní pracovať ďalej a vypočítať maticu vzdialeností. Pre správne fungovanie algoritmu budeme s dátami pracovať vo forme matice, pričom jednotlivé časové rady budú v riadkoch matice a ich hodnoty v stĺpcoch.

2. **Výpočet matice vzdialeností** - pre každé dva časové rady vypočítame vzdialenosť medzi nimi pomocou jednej z mier vzdialenosti spomínaných v podkapitole 1.2 a zostrojíme maticu vzdialeností D.

3. **Zostrojenie siete** - z matice vzdialeností zostrojíme sieť podľa výberu algoritmu z 1.3, kde vrcholmi budú jednotlivé časové rady a hrany medzi vrcholmi vzniknú na základe matice vzdialeností.

4. **Aplikovanie algoritmov detekcie komunít v sociálnych sieťach** - pomocou algoritmov detekcie komunít z 1.4 nájdeme skupiny vrcholov, ktoré majú spoločné vlastnosti a mali by patriť do spoločnej komunity.

Súčasťou článku [8] je aj naprogramovaný tento základný algoritmus [9], spolu s algoritmami vytvorenia komunít k-nn a ε-nn a funkciou normalizovania matice vzdialeností, ktoré využívame v našom rozšírenom algoritme.

V našej práci tento algoritmus rozširujeme o výpočet zhôd algoritmov, pričom pod zhodou rozumieme prípad, kedy jednotlivé algoritmy zaradia dva časové rady do rovnakého zhluku. Na základe vypočítaných zhôd jednotlivých algoritmov opäť zostrojíme sieť, pričom hrana medzi vrcholmi vznikne, ak jednotlivé časové rady boli zaradené do spoločného zhluku. Zároveň môžeme meniť hranicu existencie hrany na základe toho, koľkokrát boli časové rady zaradené do spoločného zhluku. V poslednom kroku zostrojíme klasifikátor časových radov.

2.1 Výpočet zhôd algoritmov

Na výpočet zhôd algoritmov je potrebné, aby sme algoritmus z článku [8] spustili niekoľkokrát s rôznymi nastaveniami jednotlivých parametrov. To znamená, že môžeme meniť metódu výpočtu miery vzdialenosti, algoritmus zostrojenia siete alebo algoritmus detekcie komunít v sieťach. Tieto parametre meníme vždy po uvážení vlastností konkrétnych dát. Výsledkom každého zbehnutia algoritmu je štvorcová matica, ktorej rozmer závisí od počtu časových radov v datasete. Pokiaľ časové rady X a Y sú v spoločnom zhluku, matica obsahuje na príslušnej pozícii číslo 1. Naopak, pokiaľ nie sú v spoločnom zhluku, matica na príslušnej pozícii obsahuje číslo 0. Túto maticu vypočítame pre každý zo spustených algoritmov a následne tieto matice sčítame. Dostávame maticu, ktorá sumarizuje, koľkokrát sa časové rady dostali do spoločných zhlukov.

2.2 Vážená sieť na základe zhôd algoritmov

Na základe súčtovej matice zhôd algoritmov vytvoríme váženú sieť. Vrcholmi tejto siete sú opäť časové rady. Hrany medzi vrcholmi vzniknú na základe toho, koľkokrát boli dané časové rady v spoločnom zhluku, pričom sa zaoberáme zmenami hranice existencie hrany. Cieľom je nájsť takú hranicu existencie hrany, ktorá najlepšie opisuje skutočné rozdelenie časových radov do jednotlivých komunít.

2.3 Zostrojenie klasifikátora

Po vytvorení vážených sietí z trénovacích dát, vyberieme sieť s takou hranicou existencie hrany, ktorá najlepšie opisuje skutočné rozdelenie časových radov do zhlukov. Toto rozdelenie dát budeme považovať za „správne“ a na základe tohoto rozdelenia následne klasifikujeme testovacie dáta.

Postupne vyberieme časové rady z testovacej vzorky dát a skúmame, ako sa zmení modularita, ak pridáme časový rad do jednotlivých zhlukov. Následne časový rad zaradíme do toho zhluku, pri ktorom je modularita najväčšia. Úspešnosť klasifikátora meriame pomerom správne zaradených časových radov z testovacej vzorky k celkovému počtu časových radov vo vzorke.

3 Aplikácia algoritmu na testovacie dáta

Algoritmus, podobne ako v článku [8], z ktorého vychádzame, aplikujeme na testovacie dáta. Dáta pochádzajú z UCR Time Series Classification Archive [12]. Všetky dáta obsahujú testovaciu a tréningovú vzorku a aj informáciu o tom, do akého zhluku jednotlivé časové rady v datasete patria. Algoritmus sme aplikovali na vybrané sady dát. Na podrobnejšiu ilustráciu algoritmu uvádzame detailné výsledky pre datasety *coffee* a *gunpoint*. Výsledky analýzy ďalších datasetov zhŕňame v podkapitole 3.3.

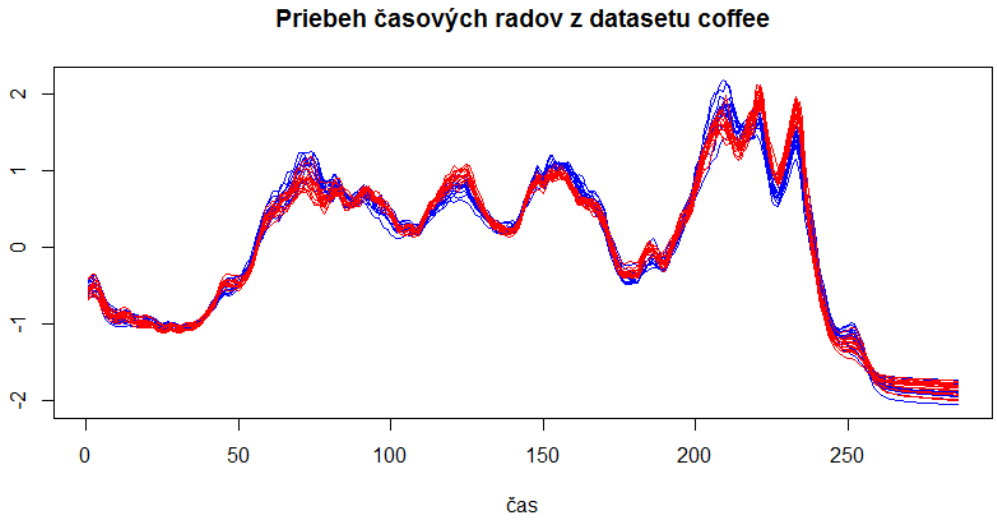
3.1 Dataset *coffee*

Dáta v datasete *coffee* pochádzajú z tzv. potravinového spektrografu. Potravinový spektrograf je prístroj, ktorý slúži na klasifikáciu potravín, skúmanie zloženia potravín a využíva sa pri hodnotení ich bezpečnosti a kvality. Spektrograf svieti na skúmaný objekt infračerveným žiarením, vďaka čomu začnú vibrovať jednotlivé molekuly a odrážať naspäť lúče svetla. Tieto lúče svetla sa zozbierajú a prejdú spektrografom, ktorý oddelí jednotlivé vlnové dĺžky svetla. Následnou analýzou získaných dát sa dá určiť, z čoho sa skúmaný objekt skladá [23].

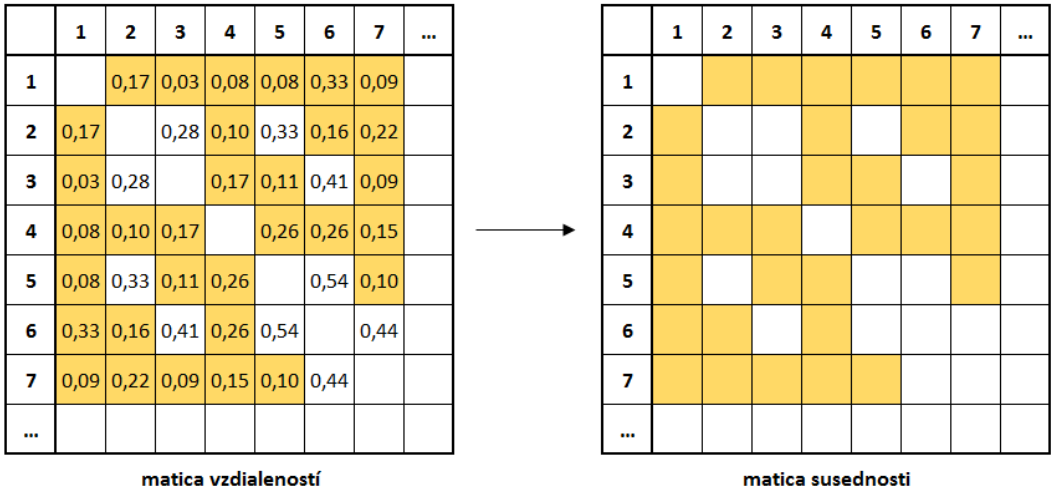
Dataset *coffee* obsahuje dáta o dvoch typoch kávových zŕn - Arabica a Robusta - obsahuje teda dve skupiny dát. Tréningová aj testovacia vzorka obsahujú po 28 časových radov. Priebeh jednotlivých časových radov z tréningovej vzorky je znázornený na Obr. 3.1, pričom červenou farbou je znázornená prvá skupina dát a modrou farbou druhá skupina dát.

Na výpočet vzdialenosti medzi časovými radmi sme zvolili euklidovskú normu. Túto normu sme zvolili kvôli tomu, že jednotlivé časové rady sa nelíšia v rýchlosti, preto sú tzv. *lock-step* miery vhodné na meranie vzdialenosti takýchto časových radov. Sieť sme vytvorili pomocou algoritmu k-nn s $k = 8$. Každý časový rad teda spojíme s ôsmimi najbližšími časovými radmi. Vytvorenie matice susednosti z matice vzdialeností sme znázornili na Obr. 3.2.

Následne sme aplikovali všetky spomenuté algoritmy detekcie komunít v sieťach, ktorých výsledky sme znázornili na Obr. 3.3. Jednotlivé komunity sú odlíšené farbami vrcholov, čísla vrcholov označujú zaradenie do skupín, ktoré máme dostupné z dát -

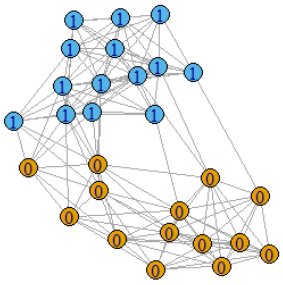


Obr. 3.1: Priebeh časových radov z datasetu *coffee*

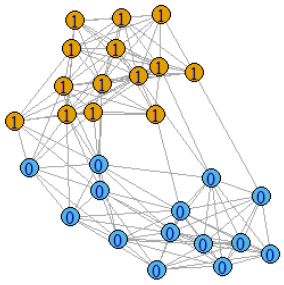


Obr. 3.2: Ilustrácia vytvorenia matice susednosti z matice vzdialeností pomocou algoritmu k-nn s $k = 8$ z datasetu *coffee*. V matici vzdialeností sú farebne vyznačené najkratšie vzdialenosti pre každý vrchol, na vyznačených miestach v matici susednosti budú jednotky (medzi vrcholmi vznikne hrana) a inde nuly (medzi vrcholmi nevznikne hrana)

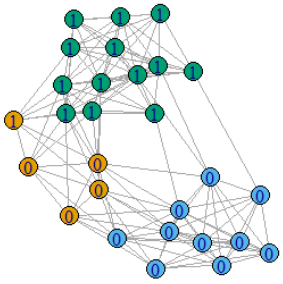
nulou je označená prvá skupina dát, jednotkou druhá skupina dát. Z Obr. 3.3 vidíme, že algoritmy *walktrap*, *fast & greedy* a *infomap* zaradili všetky časové rady správne - časové rady zo skupiny 0 do jedného zhluku a časové rady zo skupiny 1 do druhého zhluku. Algoritmus *label propagation* zaradil nesprávne dva časové rady a algoritmus *multilevel* zaradil dáta do troch zhlukov.



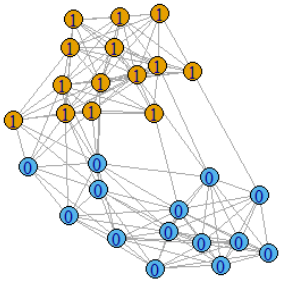
(a) algoritmus *walktrap*



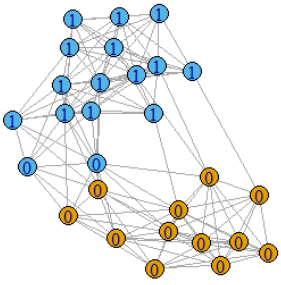
(b) algoritmus *fast & greedy*



(c) algoritmus *multilevel*



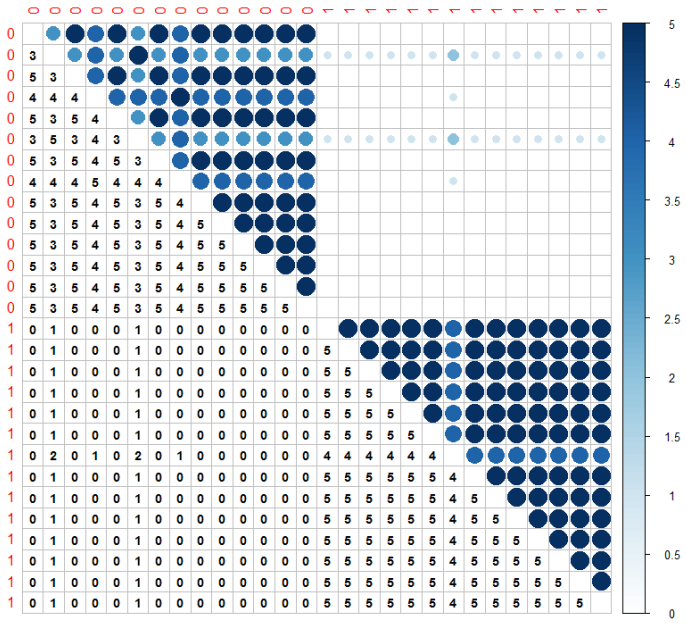
(d) algoritmus *infomap*



(e) algoritmus *label propagation*

Obr. 3.3: Ilustrácia aplikácie algoritmov detekcie komunít na sieť *coffee*

Následne sme spočítali zhody algoritmov - teda koľkokrát sa dva časové rady dostali do spoločného zhluku. Výsledkom je matica S, ktorá je znázornená na Obr. 3.4. Keďže

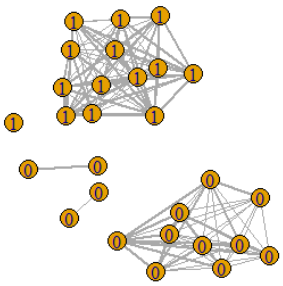


Obr. 3.4: Ilustrácia matice S z datasetu *coffee*

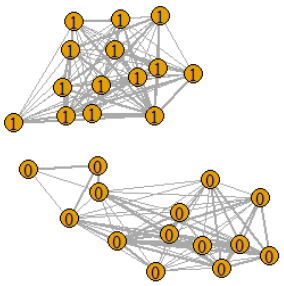
dáta v datasete *coffee* sú usporiadané tak, že prvých 14 časových radov patrilo do skupiny 0 a druhých 14 časových radov do skupiny 1, môžeme na Obr. 3.4 vidieť akoby dva trojuholníky, ktoré značia, že väčšina algoritmov (teda 3 z 5) zaradila všetky časové rady správne.

Na základe matice S sme zostrojili siete, pri ktorých existencia hrany závisí od hranice, ktorú zvolíme na základe pozorovania. Pod hranicou rozumieme, koľkokrát dva algoritmy zaradili dva časové rady do spoločného zhluku, teda prvky matice S. Siete pre jednotlivé hranice existencie hrany v sieťach sú zobrazené na Obr. 3.5.

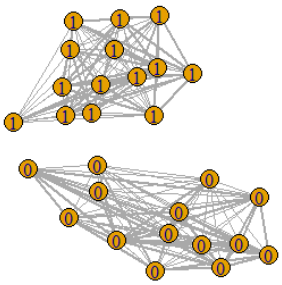
Na Obr. 3.5 môžeme vidieť rozpad na jednotlivé komunity na základe výberu hranice existencie hrán. Môžeme si všimnúť, že pokiaľ za hranicu existencie hrany zvolíme aspoň 4 alebo aspoň 3 zhody, dostaneme sieť s dvomi komponentami, pričom všetky časové rady sú zaradené podľa rozdelenia, ktoré máme k dispozícii z dát. Za hranicu existencie hrany vezmeme teda aspoň 3 zhody, pretože toto rozdelenie komunít má vyššiu modularitu (0,5000) ako rozdelenie s hranicou existencie hrany aspoň 4 zhody (0,4923) a túto sieť využijeme pri tvorbe klasifikátora, pretože predpokladáme, že by



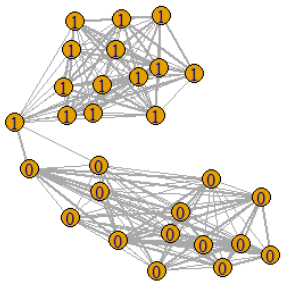
(a) hranica existencie hrany aspoň 5 zhôd



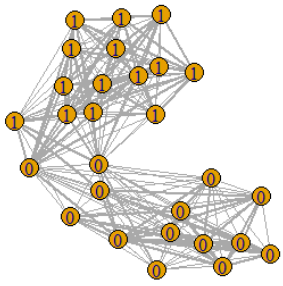
(b) hranica existencie hrany aspoň 4 zhody



(c) hranica existencie hrany aspoň 3 zhody



(d) hranica existencie hrany aspoň 2 zhody



(e) hranica existencie hrany aspoň 1 zhoda

Obr. 3.5: Vážené siete z matice S z datasetu *coffee* pre rôzne hranice existencie hrany, váha hrany je znázornená jej hrúbkou

sme pomocou nej mohli zaradiť väčšinu časových radov z testovacej vzorky správne.

Vezmeme prvý časový rad z testovacej vzorky a vytvoríme novú sieť s hranicou existencie hrany aspoň 3 zhody, ktorá bude obsahovať tento časový rad. Následne pridávame časový rad do jednotlivých zhlukov a vypočítame modularitu. Nakoniec zaradíme časový rad do toho zhluku, pri ktorom je modularita najväčšia.

Aplikáciou tohoto postupu na testovaciu vzorku dát *coffee* sme dostali výsledky zhrnuté v Tabuľke 3.1. V prvom stĺpci sú poradové čísla časových radov, v stĺpci s názvom Pôvodné zaradenie je zaradenie časových radov dostupné z dát, v ďalšom stĺpci zaradenie získané z klasifikátora. V stĺpci Modularita 0 je vypočítaná modularita, keby bol časový rad priradený do komunity, kde sú časové rady skupiny 0, a Modularita 1 je vypočítaná modularita v prípade, kedy by bol časový rad priradený do komunity s časovými radmi zo skupiny 1. Z výsledkov vidíme, že klasifikátor zaradil správne všetky časové rady okrem jedného, čo predstavuje $96,43\%$-nú úspešnosť zostrojeného klasifikátora.

3.2 Dataset *gunpoint*

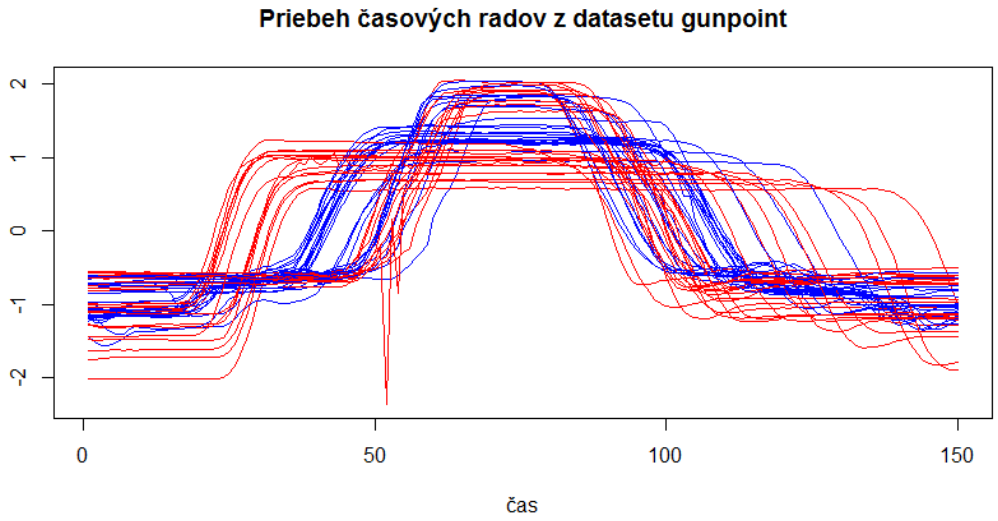
Podobne, ako v prípade dát *coffee*, aj dáta *gunpoint* pochádzajú z UCR Time Series Classification Archive [12]. Tento dataset obsahuje dáta o jednom mužovi a jednej žene, ktorí robia pohyb rukou. Dáta sú rozdelené na dve skupiny časových radov „Gun-Draw“ a „Point“. Pri Gun-Draw dátach majú protagonisti ruky pri tele, vyberú zbraň z puzdra, ktoré majú umiestnenú na svojom boku a na sekundu so zbraňou namieria na cieľ. Pri dátach Point majú protagonisti opäť ruky pri tele a opäť zamieria na cieľ, avšak s tým rozdielom, že budú mieriť len svojim ukazovákom. Pre obe skupiny dát bolo zmerané ťažisko pravej ruky protagonistov v smere osi x a y. Dáta obsahujú informáciu len o osi x.

Trénovacia vzorka dát obsahuje 50 časových radov, v každej skupine je 25 časových radov. Priebeh časových radov je zobrazený na Obr. 3.6, pričom dáta sú rozdelené farebne podľa skupín, do ktorých patria. Z Obr. 3.6 je vidieť, že dáta v rámci skupín sú značne rozdielne. Navyše sa zdá, že každá skupina dát sa akoby delí na ďalšie, čo bude mať vplyv aj na našu analýzu.

Na meranie vzdialenosti medzi časovými radmi sme vybrali mieru Dynamic Time

Por. č.	Pôvodné zaradenie	Zaradenie z klasifikátora	Modularita 0	Modularita 1
1	0	0	0.3818	0.3650
2	0	0	0.4974	0.4286
3	0	0	0.4915	0.4414
4	0	0	0.4974	0.4286
5	0	0	0.4924	0.4239
6	0	0	0.4974	0.4286
7	0	0	0.3862	0.3793
8	0	0	0.4974	0.4286
9	0	0	0.4994	0.4266
10	0	0	0.4974	0.4286
12	0	0	0.4924	0.4239
13	0	0	0.4932	0.4156
14	0	0	0.4994	0.4266
15	0	0	0.4974	0.4286
16	1	1	0.4286	0.4974
17	1	1	0.4286	0.4974
18	1	1	0.4286	0.4974
19	1	1	0.3311	0.3807
20	1	1	0.4286	0.4974
21	1	1	0.4286	0.4974
22	1	0	0.3706	0.3685
23	1	1	0.4286	0.4974
24	1	1	0.4286	0.4974
25	1	1	0.4286	0.4974
26	1	1	0.3624	0.4135
27	1	1	0.4286	0.4974
28	1	1	0.4286	0.4974

Tabuľka 3.1: Výsledky klasifikátora aplikovaného na dataset *coffee*

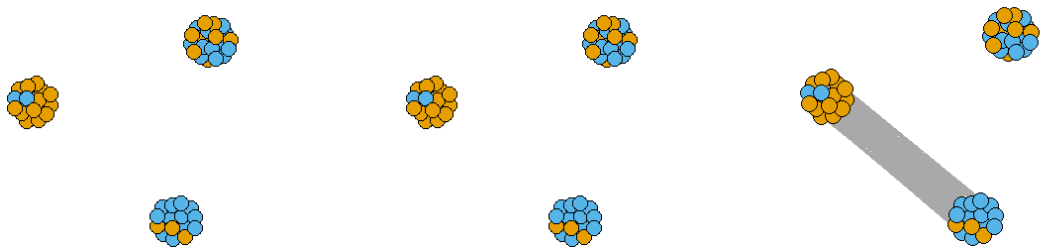


Obr. 3.6: Priebeh časových radov z datasetu *gunpoint*

Warping (DTW) kvôli tomu, že z Obr. 3.6 sa zdá, že medzi časovými radmi vrámci jednej skupiny je prítomný aj posun v čase. Sieť sme vytvorili pomocou algoritmu k-nn s $k = 12$. Parameter k sme zvolili na základe pozorovania priebehu dát. Keďže dáta obsahujú dve triedy dát, každá trieda po 25 časových radov, chceme, aby bol každý vrchol spojený najviac s 24 ďalšími vrcholmi. Tiež sme si všimli, že dáta v každej triede obsahujú akoby 2 ďalšie podtriedy na základe ich priebehu. Teda každý vrchol by mal byť spojený len s približne polovicou dát zo svojej triedy.

Algoritmus s takýmto nastavením parametrov sme spustili pre všetkých 5 algoritmov zhlukovania a vypočítali sme, koľkokrát sa tieto algoritmy zhodli pri zhlukovaní. Následne, sme podobne ako pri dátach *coffee* zostrojili vážené siete znázornené na Obr. 3.7, pôvodné zaradenie do tried je znázornené farebne.

Na Obr. 3.7a a Obr. 3.7b je vidno vytvorenie troch komunít, pričom dve z nich obsahujú prevažne časové rady z jednej triedy. Jedna komunita je tzv. zmiešaná, obsahujúca časové rady z oboch tried. Pokiaľ znížime hranicu existencie hrany na aspoň 3 zhody, môžeme si všimnúť, že sa dve komunity, ktoré obsahujú časové rady prevažne z jednej triedy spoja a vytvoria tak ďalšiu zmiešanú komunitu. Takéto delenie na 2 komunity nemôžeme použiť na klasifikáciu, pretože by sme dostali neuspokojivé výsledky a je pravdepodobné, že úspešnosť klasifikátora by bola na úrovni 50%, teda klasifikátor by rozhodoval náhodne. Vezmeme teda sieť s hranicou existencie hrany aspoň 4



(a) hranica existencie hrany aspoň 5 zhôd

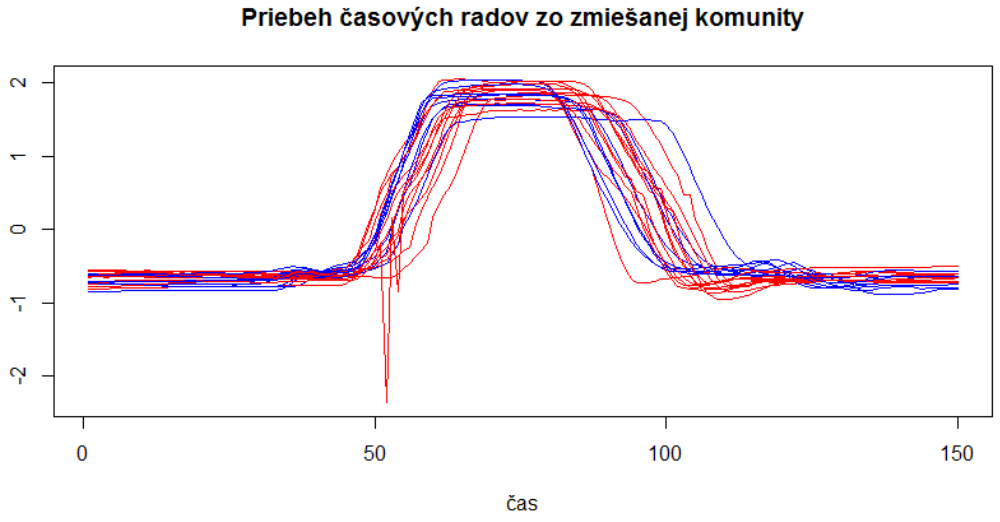
(b) hranica existencie hrany aspoň 4 zhody

(c) hranica existencie hrany aspoň 3 zhody

Obr. 3.7: Vážené siete z matice S z z datasetu *gunpoint* pre rôzne hranice existencie hrany

zhody, vytvoríme klasifikátor, ktorý testovacie dáta rozdelí na 3 zhluky a časové rady zo zmiešanej komunity budeme ďalej deliť. V tomto prípade sú vážené siete s hranicou existencie hrany aspoň 5 zhôd a aspoň 4 zhody rovnaké, pretože matica S, z ktorej sme tieto siete vytvárali obsahovala iba hodnoty 5 a 3.

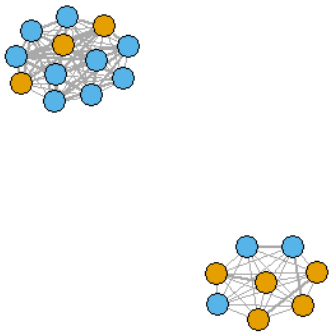
Priebeh časových radov zo zmiešanej komunity sme zobrazili na Obr. 3.8. Je zrejmé,



Obr. 3.8: Priebeh časových radov zo zmiešanej komunity z datasetu *gunpoint*

že dáta sú medzi sebou výrazne podobné a preto boli pri prvom zhlukovaní zaradené do spoločnej komunity. Zopakujeme teda postup vytvorenia siete ako v predchádzajú-

com prípade, parameter k zvolíme rovný 5. Následne opäť aplikujeme všetky algoritmy detekcie komunít a spočítame, koľkokrát sa algoritmy zhodli pri vytváraní komunít. V našom prípade sa všetky algoritmy zhodli pri všetkých časových radoch, dostávame teda iba jednu váženú sieť, ktorá je zobrazená na Obr. 3.9. Je zrejmé, že ani takéto roz-

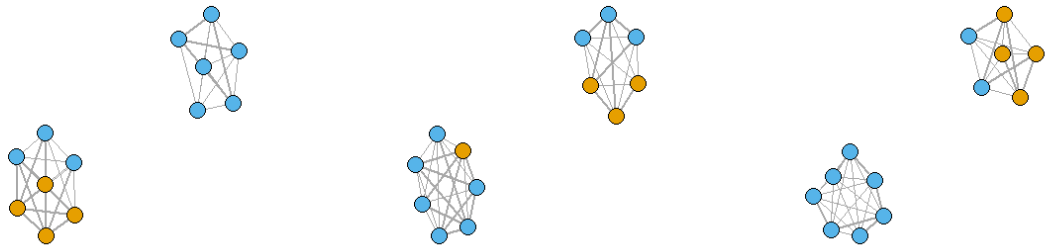


Obr. 3.9: Vážená sieť z dát zo zmiešanej komunity z datasetu *gunpoint*

delenie dát by nám pri klasifikácii zmiešaného zhluku neposkytlo uspokojivé výsledky, kvôli tomu, že až 6 časových radov z 20-tich je zaradených nesprávne a vytvorené sú 2 menšie zmiešané zhluky. Dôvodom takýchto nesprávne vytvorených komunít môže byť fakt, že časové rady nie sú „rovnako rozdielne“ v celej ich dĺžke, ale len v ich časti. Rozhodli sme sa preto na ďalšiu klasifikáciu použiť len časť z dát.

Aby sme sa vyhli prípadnému overfittingu, pri rozhodovaní, ktorú časť z dát použiť, rozdelili sme trénovacie dáta zo zmiešaného zhluku na trénovaciu a validačnú časť. Náhodným výberom sme sa snažili dáta rozdeliť približne v pomere 70:30. Z 20 časových radov bolo vybraných 13 časových radov na trénovanie a 7 časových radov na validáciu. Intuitívne sa nám zdalo, že najväčšie rozdiely v dátach sú v strednej časti časových radov. Zobrali sme teda postupne začiatočnú časť trénovacej časti dát (prvých 40 hodnôt), strednú časť trénovacej časti dát (hodnoty od 41 do 110) a poslednú časť trénovacej časti dát (posledných 40 hodnôt) a z každej časti dát sme vytvorili sieť s parametrom $k = 3$, aplikovali sme algoritmy detekcie komunít a vytvorili vážené siete. Cieľom bolo nájsť váženú sieť s takou hranicou existencie hrany, ktorá by najlepšie

popisovala skutočné rozdelenie dát - teda má dva komponenty a v jednotlivých komponentoch sú dáta z jednej triedy. Túto váženú sieť sme následne použili pri vytváraní klasifikátora. Vážené siete, použité pri tvorbe klasifikátora, sú pre jednotlivé časti dát zobrazené na Obr. 3.10.



(a) prvých 40 hodnôt dát **(b)** hodnoty od 41 do 110 **(c)** posledných 40 hodnôt dát

Obr. 3.10: Vážené siete z trénovacej časti trénovacích dát pre jednotlivé úseky dát z datasetu *gunpoint* použité pri tvorbe klasifikátorov

Klasifikátory, vytvorené z vážených sietí pre jednotlivé časové úseky trénovacej časti trénovacích dát, sme aplikovali na validačnú časť trénovacích dát. Cieľom bolo zistiť presnosť klasifikátorov použitých na jednotlivé časové úseky validačných dát a tým zistiť, ktorý časový úsek testovacích dát použiť na klasifikáciu. Výsledky klasifikácie validačných dát sú zobrazené v Tabuľkách 3.2, 3.3 a 3.4. Na základe toho, koľko časových radov z validačných dát bolo zaradených správne sme vypočítali presnosť jednotlivých klasifikátorov a výsledky sme zhrnuli v Tabuľke 3.5.

Z výsledkov zhrnutých v Tabuľke 3.5 vidíme, že najvyššiu presnosť klasifikácie sme dosiahli pri použití posledných 40 hodnôt, preto pri tvorbe klasifikátora použijeme sieť z týchto hodnôt, ktorá je zobrazená na Obr. 3.10c.

Pri klasifikovaní testovacích dát sme aplikovali klasifikátor vytvorený v prvom kroku. Výsledkom je rozdelenie testovacích dát na 3 triedy. Následne na dáta, ktoré boli klasifikované do zmiešaného zhluku sme aplikovali klasifikátor vytvorený z trénovacích dát zo zmiešaného zhluku. Najvyššiu úspešnosť dosiahol klasifikátor vytvorený z posledných 40 hodnôt trénovacích dát a preto sme aj pri klasifikácii testovacích dát použili

Por. č.	Pôvodné zaradenie	Zaradenie z klasifikátora	Modularita 1	Modularita 2
1	1	1	0.4543	0.3275
2	2	2	0.3469	0.5000
3	2	2	0.3469	0.5000
4	1	1	0.4543	0.3275
5	1	2	0.3469	0.5000
6	2	2	0.3469	0.5000
7	1	2	0.3469	0.5000

Tabuľka 3.2: Výsledky klasifikátora aplikovaného na prvých 40 hodnôt validačnej časti dát *gunpoint*

Por. č.	Pôvodné zaradenie	Zaradenie z klasifikátora	Modularita 1	Modularita 2
1	1	1	0.4882	0.4430
2	2	2	0.3251	0.4915
3	2	2	0.3251	0.4915
4	1	2	0.3342	0.4974
5	1	2	0.3251	0.4915
6	2	2	0.3047	0.4688
7	1	2	0.3642	0.4938

Tabuľka 3.3: Výsledky klasifikátora aplikovaného na strednú časť hodnôt validačnej časti dát *gunpoint*

posledných 40 hodnôt.

Z celkového počtu 150 testovacích časových radov sme v prvom kroku klasifikácie dosiahli výsledky zhrnuté v Tabuľke 3.6. Správne bolo zaradených 54 časových radov, 80 bolo zaradených do zmiešaného zhluku a 16 časových radov bolo zaradených nesprávne. Pri ďalšej klasifikácii testovacích dát a použití klasifikátora vytvoreného z posledných 40 hodnôt dát sme dosiahli výsledky zhrnuté v Tabuľke 3.7. Správne bolo zaradených 71 časových radov, 9 bolo zaradených nesprávne.

Por. č.	Pôvodné zaradenie	Zaradenie z klasifikátora	Modularita 1	Modularita 2
1	1	1	0.4184	0.2494
2	2	2	0.3462	0.3780
3	2	2	0.3304	0.4995
4	1	2	0.3017	0.3130
5	1	1	0.5000	0.3469
6	2	2	0.2739	0.3878
7	1	1	0.3520	0.2771

Tabuľka 3.4: Výsledky klasifikátora aplikovaného na posledných 40 hodnôt validačnej časti dát *gunpoint*

Časový úsek validačných dát	Počet správne zaradených časových radov	Percentuálna úspešnosť klasifikácie
prvých 40 hodnôt	5	71,43%
hodnoty od 41 do 110	4	57,14%
posledných 40 hodnôt	6	85,71%

Tabuľka 3.5: Úspešnosť klasifikátorov aplikovaných na jednotlivé časové úseky validačných dát *gunpoint*

Typ zhluku	Počet zaradených časových radov	Počet správne zaradených časových radov
zhluk s prevažným počtom dát zo skupiny 1	30	24
zhluk s prevažným počtom dát zo skupiny 2	40	30
zmiešaný zhluk	80	-

Tabuľka 3.6: Zaradenie časových radov v prvom kroku klasifikácie datasetu *gunpoint*

Celkovo sa nám teda podarilo úspešne klasifikovať 125 zo 150 časových radov, čo predstavuje úspešnosť klasifikácie na úrovni 83,33%.

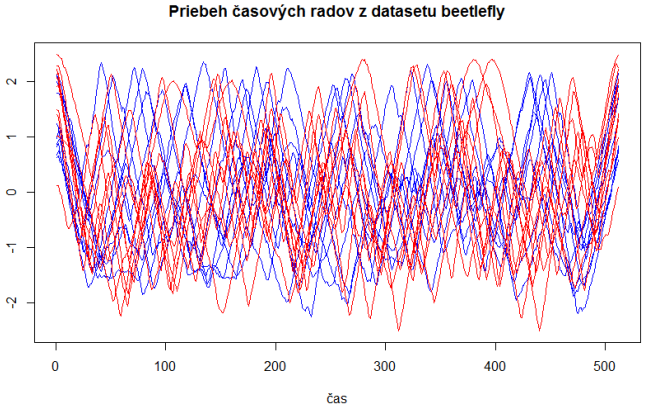
Typ zhluku	Počet zaradených časových radov	Počet správne zaradených časových radov
zhluk s prevažným počtom dát zo skupiny 1	41	37
zhluk s prevažným počtom dát zo skupiny 2	39	34

Tabuľka 3.7: Zaradenie časových radov v druhom kroku klasifikácie datasetu *gunpoint*

3.3 Ďalšie datasety

Algoritmus spolu s klasifikátorom sme aplikovali aj na ďalšie datasety z UCR Time Series Classification Archive [12]. Pripomeňme, že pri vytváraní vážených sietí, počítame zhody po použití piatich algoritmov zhlukovania. Vo všetkých prípadoch sme použili algoritmus, ktorý bol použitý aj pri datasete *coffee*, teda vytvorenie len jednej siete, na základe ktorej bol vytvorený klasifikátor. Výsledky klasifikátora spolu so zvolenými parametrami sme zobrazili v Tabuľke 3.8.

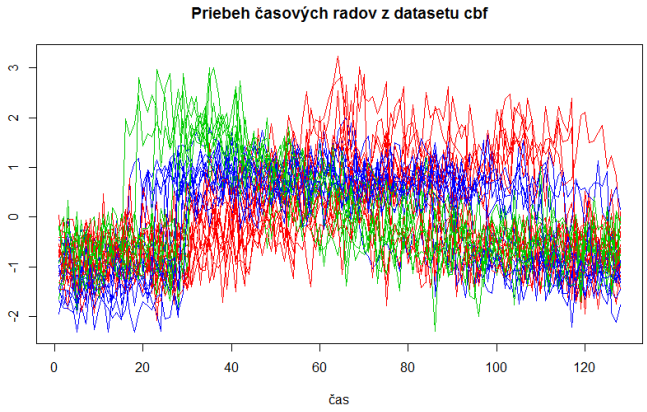
- *beetlefly* sú dáta, ktoré vznikli z obrysov obrázkov, pričom obrysy boli transformované do tvaru časového radu. Cieľom je podľa ich obrysu rozlíšiť, či sa na obrázku nachádzala mucha alebo chrobák. Priebeh dát je zobrazený na Obr. 3.11.



Obr. 3.11: Priebeh časových radov z datasetu *beetlefly*

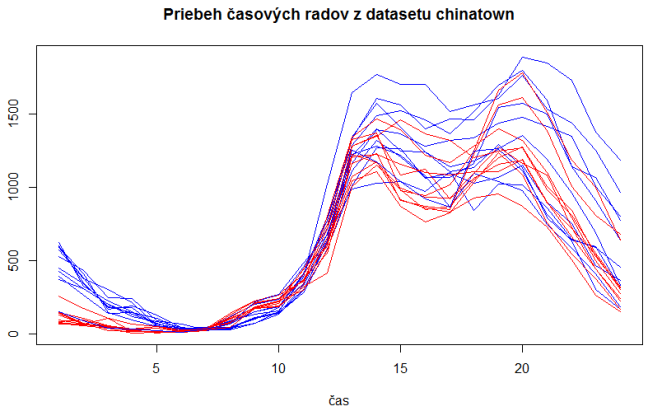
- *cbf* alebo Cylinder-Bell-Funnel sú simulované dáta. Dataset sa skladá z troch tried dát, každá z nich je normálny biely šum, líšia sa v posune. Priebeh dát je

zobrazený na Obr. 3.12.



Obr. 3.12: Priebeh časových radov z datasetu *cbf*

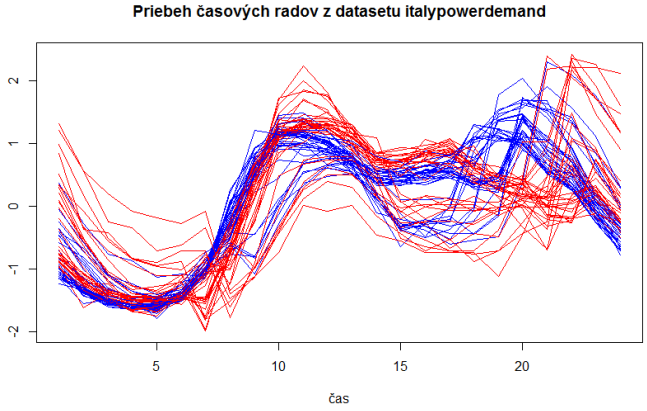
- *chinatown* sú dáta, ktoré pochádzajú z automatického systému rátania chodcov v Austrálii v Chinatown - Swanston Street. Dáta sú rozdelené na 2 triedy podľa toho, či ide o dáta z víkendu alebo z pracovného dňa. Priebeh dát je zobrazený na Obr. 3.13.



Obr. 3.13: Priebeh časových radov z datasetu *chinatown*

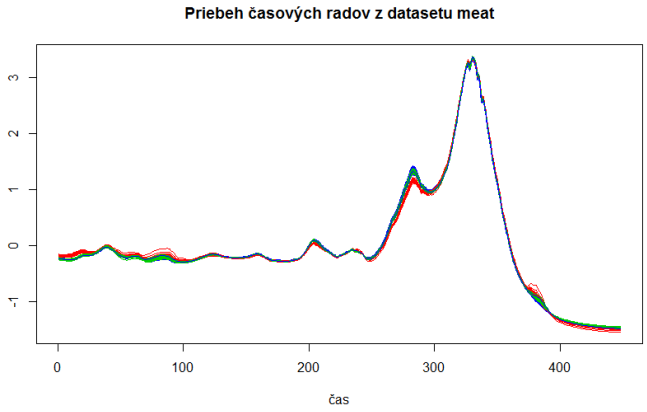
- *italypowerdemand* sú dáta o mesačnom dopyte po elektrickej energii v Taliansku. Časové rady sú rozdelené na 2 triedy podľa toho, či ide o dáta z augusta alebo zo zvyšku roka. Priebeh dát je zobrazený na Obr. 3.14.

- *meat* sú dáta o druhoch mäsa, ktoré podobne ako dáta *coffee* pochádzajú z potravinového spektrografu. Dáta obsahujú 3 triedy, každá trieda dát reprezentuje



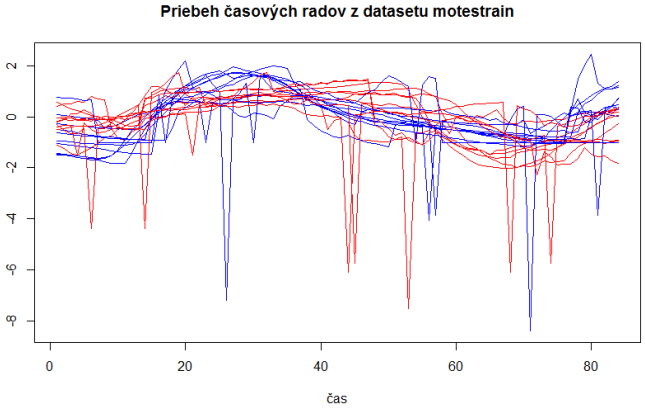
Obr. 3.14: Priebeh časových radov z datasetu *italypowerdemand*

jeden druh mäsa - kuracie, morčacie a bravčové mäso. Priebeh časových radov je zobrazený na Obr. 3.15.

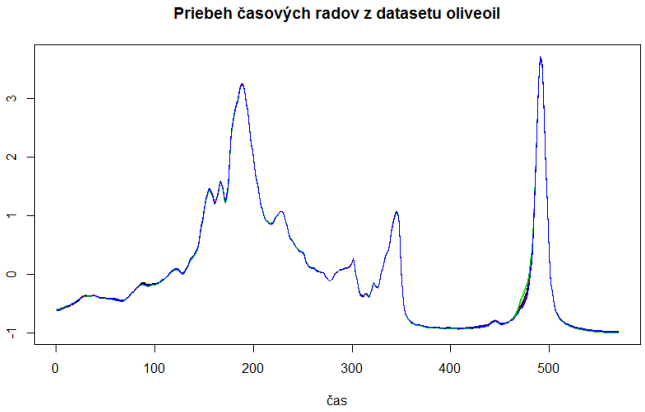


Obr. 3.15: Priebeh časových radov z datasetu *meat*

- *motestrain* sú dáta pochádzajúce zo senzorov, pričom v jednej triede dát sú časové rady reprezentujúce vlhkosť a v druhej triede sú časové rady reprezentujúce teplotu v nejakom prostredí. Priebeh časových radov je zobrazený na Obr. 3.16.
- *oliveoil* sú dáta o olivovom oleji, ktoré podobne ako dáta *coffee* pochádzajú z potravinového spektrografu. Dáta obsahujú 4 triedy, v každej z nich ide o extra panenský olivový olej. Triedy sa líšia krajinou pôvodu olivového oleja. Priebeh dát je zobrazený na Obr. 3.17.

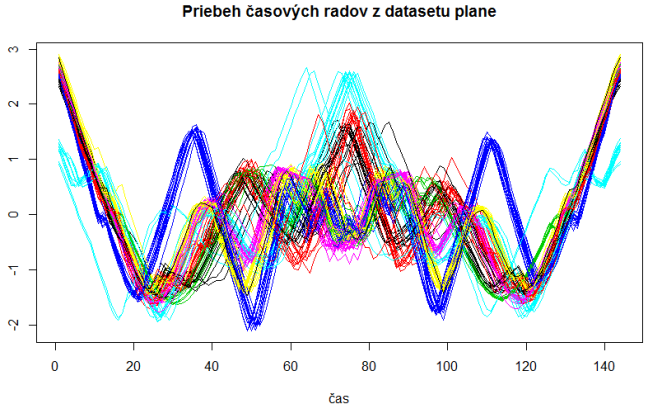


Obr. 3.16: Priebeh časových radov z datasetu *motestrain*



Obr. 3.17: Priebeh časových radov z datasetu *oliveoil*

- *plane* sú dáta o obrysoch lietadiel. Dataset obsahuje 7 tried, každá z nich reprezentuje jedno z lietadiel - Mirage, Eurofighter, F-14 (sklopené krídla), F-14 (roztvorené krídla), Harrier, F-22 a F-15. Priebeh dát je zobrazený na Obr. 3.18.



Obr. 3.18: Priebeh časových radov z datasetu *plane*

Názov datasetu	Zvolená vzdialenosť	Metóda vytvorenie siete	Hranica existencie hrany	Úspešnosť klasifikátora
beetlefly	DTW	k-nn, $k = 4$	aspoň 2 zhody	70,00%
cbf	DTW	ε-nn, $\varepsilon = 0,3$	aspoň 3 zhody	94,11%
chinatown	euklidovská	ε-nn, $\varepsilon = 0,3$	aspoň 3 zhody	71,01%
italypowerdemand	euklidovská	k-nn, $k = 20$	aspoň 4 zhody	87,07%
meat	euklidovská	k-nn, $k = 7$ a $k = 8$	aspoň 5 zhôd	83,33%
motestrain	euklidovská	k-nn, $k = 4$	aspoň 5 zhôd	83,55%
oliveoil	korelácia	k-nn, $k = 2$ a $k = 3$	aspoň 5 zhôd	76,67%
plane	euklidovská	k-nn, $k = 7$	aspoň 4 zhody	95,24%

Tabuľka 3.8: Výsledky klasifikátora aplikovaného na ďalšie datasety

4 Aplikácia algoritmu na reálne dáta

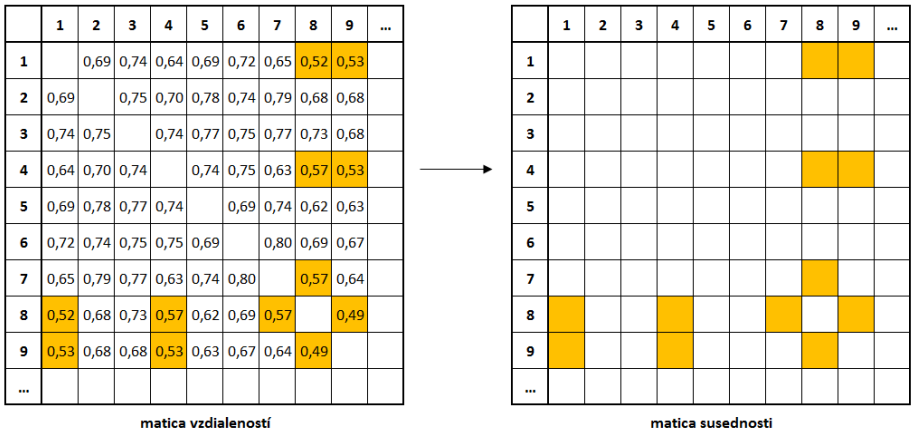
Algoritmus sme okrem vybratých testovacích dát, ktoré boli použité v článku [8], aplikovali aj na reálne dáta. Zvolili sme denné výnosy akcií indexu S&P 500 za rok 2018.

Index S&P 500 (Standard & Poor's 500) je indexom najväčších (na základe hodnoty) amerických spoločností, ktoré sú obchodované na burzách NYSE a NASDAQ. Tento index je často opisovaný ako index, ktorý najlepšie zastupuje americký akciový trh. Dáta sme stiahli pomocou knižnice BatchGetSymbols [21] priamo v softvéri R, pričom sme zvolili obdobie od 1. januára 2018 do 31. decembra 2018. Niektoré dáta neboli z rôznych dôvodov kompletné a obsahovali NA hodnoty. Na dopočítanie týchto hodnôt sme využili funkcie implementované v knižnici imputeTS [18]. Zvolili sme funkciu `na_kalman` s parametrom `model=auto.arima`, ktorá na dopočítanie chýbajúcich hodnôt používa tzv. Kalmanove zhladzovanie.

Pracovali sme s datasetom 493 časových radov, každý časový rad pozostáva z 250 hodnôt. K dispozícii máme aj informáciu, do akého odvetvia akcia patrí [15]. Keďže pri takomto type dát, nevieme určiť aké je „správne“ rozdelenie časových radov do zhlukov, budeme naše výsledky porovnávať práve s rozdelením výnosov akcií podľa odvetvia, do ktorého patria. Môžeme predpokladať, že napriek tomu, že priebehy cien akcií sa riadia náhodnými procesmi, akcie z rovnakého odvetvia budú mať podobný priebeh výnosov počas roka, pretože ich ovplyvňujú viaceré spoločné faktory, akými sú napríklad sezónnosť alebo fáza ekonomického cyklu.

Maticu vzdialeností sme vypočítali pomocou korelácie (1.8). Koreláciu ako vhodnú vzdialenosť sme vybrali práve z dôvodu, že pokiaľ priebehy cien akcií ovplyvňujú spoločné faktory, mali by byť tieto časové rady medzi sebou korelované. Matica vzdialeností je znormovaná a obsahuje hodnoty v intervale $[0, 1]$. Vytvorili sme sieť pomocou algoritmu ε-nn, kde hrana medzi dvomi vrcholmi vznikne vtedy, ak je ich vzájomná vzdialenosť nižšia ako hraničná hodnota ε. Hraničnú hodnotu ε sme zvolili na troch úrovniach - 0,4, 0,5 a 0,6. Vytvorenie matice susednosti z matice vzdialeností pre sieť z výnosov akcií s parametrom $\varepsilon = 0,6$ sme zobrazili na Obr. 4.1.

Na Obr. 4.2 si môžeme všimnúť, že so zvyšujúcou sa hraničnou hodnotou ε-nn sa počet hrán v sieti tiež zvyšuje, pretože máme nižšie nároky na podobnosť dvoch vrcholov. Zároveň sa znižuje počet tzv. izolovaných vrcholov, ktoré sú zrejme napriek



Obr. 4.1: Ilustrácia vytvorenia matice susednosti z matice vzdialeností pomocou algoritmu ε-nn s $\varepsilon = 0,6$ z dát výnosov akcií. V matici vzdialeností sú farebne vyznačené vzdialenosti menšie ako ε, na príslušných miestach v matici susednosti budú jednotky (medzi vrcholmi vznikne hrana) a inde nuly (medzi vrcholmi nevznikne hrana)

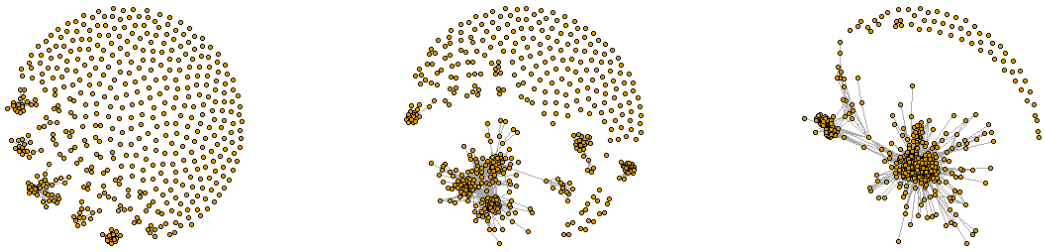
znižujúcim sa nárokom na vzdialenosť dvoch vrcholov stále veľmi vzdialené od zvyšku vrcholov.

Pre každú vytvorenú sieť sme vypočítali hodnotu modularity v prípade, že by sme v sieti vytvorili zhluky na základe odvetvia, do ktorého jednotlivé akcie patria. Hodnoty modularity takýchto zhlukovaní pre siete s rôznymi hraničnými hodnotami ε sme zhrnuli v Tabuľke 4.1. Hodnoty modularity v Tabuľke 4.1 budeme porovnávať s hodnotami modularity pre rôzne zhlukovania v tej istej sieti (v sieti vytvorenej pomocou rovnakej hraničnej hodnoty ε).

Hraničná hodnota ε	Modularita
$\varepsilon = 0,4$	0,7371
$\varepsilon = 0,5$	0,6066
$\varepsilon = 0,6$	0,2327

Tabuľka 4.1: Modularita rozdelenia sietí v prípade, že by boli siete rozdelené podľa odvetví pri rôznych hraničných hodnotách ε. Hodnoty modularity budeme porovnávať pre rôzne zhlukovania v tej istej sieti (v sieti vytvorenej pomocou rovnakej hraničnej hodnoty ε)

Na siete sme aplikovali algoritmy zhlukovania, na základe ktorých sme vytvorili vážené siete. Vo vážených sieťach považujeme komponenty za komunity a počítame



(a) sieť s $\varepsilon = 0,4$ **(b)** sieť s $\varepsilon = 0,5$ **(c)** sieť s $\varepsilon = 0,6$

Obr. 4.2: Siete z dát výnosov akcií pre rôzne hranice existencie hrany

modularitu takýchto zhlukovaní. Pripomeňme, že vážené siete vytvárame na základe zhôd algoritmov zhlukovania, ktoré počítame po aplikovaní piatich algoritmov. Mohlo by sa zdať, že stačí požadovať čo najväčší počet zhôd algoritmov a vytvoriť váženú sieť s najväčším možným počtom zhôd. Pri takomto prístupe môže nastať situácia, že v sieti vznikne veľký počet malých zhlukov, čo nemusí byť optimálne, preto pri každom zhlukovaní vypočítame jeho modularitu. Za finálne zhlukovanie budeme považovať sieť s takým počtom zhôd, pri ktorom dosiahneme najvyššiu modularitu.

Vážené siete z matice S pre dáta výnosov akcií zo siete s parametrom $\varepsilon = 0,4$ sú zobrazené na Obr. 4.3. Zobrazili sme vážené siete pre hranice existencie hrany aspoň 5 zhôd, aspoň 4 zhody a aspoň 2 zhody, pretože matica S obsahovala len tieto hodnoty zhôd. Ak za hranicu existencie hrany zvolíme aspoň 5 zhôd, vzniknuté zhluky sú síce zhlukmi s menším počtom vrcholov, avšak obsahujúce prevažne akcie z rovnakého odvetvia. Takáto sieť obsahuje veľký počet izolovaných vrcholov. Pri znižovaní hranice existencie hrany na aspoň 4 alebo aspoň 2 zhody si môžeme všimnúť vytvorenie jedného väčšieho zhluku, do ktorého sa spojili menšie zhluky obsahujúce akcie z rovnakého odvetvia. Zdá sa teda, že tieto zhlukovania sú podobné so zhlukovaním po odvetviach. Pri znižovaní hranice existencie hrany si môžeme na Obr. 4.3b a Obr. 4.3c všimnúť, že vzniká väčší zhluk vrcholov, pričom v zhluku sú opäť prevažne akcie z jedného odvetvia. Izolované vrcholy pri sieti s hranicou existencie hrany aspoň 5 zhôd zostali izolované aj po znížení nárokov na hranicu existencie hrany. Modularitu pre

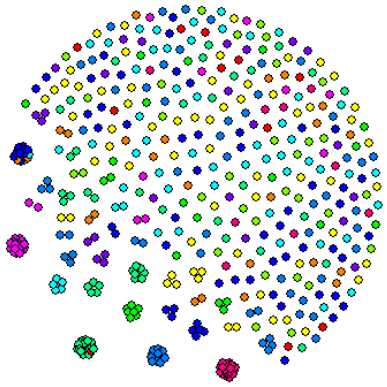
rôzne hranice existencie hrany sme zhrnuli v Tabuľke 4.2. Z Tabuľky 4.2 vidíme, že najvyššiu modularitu dosiahlo zhlukovanie v sieti s aspoň 5 zhodami.

Hranica existencie hrany	Modularita
aspoň 5 zhôd	0,8214
aspoň 4 zhody	0,7806
aspoň 3 zhody	0,7806
aspoň 2 zhody	0,7001
aspoň 1 zhoda	0,7001
rozdelenie podľa odvetví	0,7371

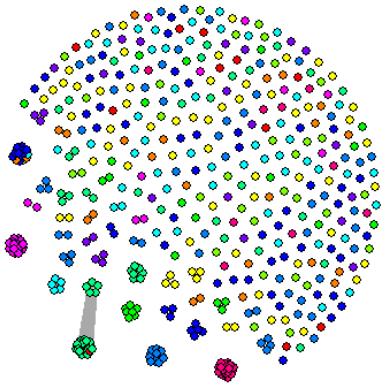
Tabuľka 4.2: Modularita na základe hranice existencie hrany vo váženej sieti a zhlukovania podľa odvetví pre sieť s parametrom $\varepsilon = 0,4$ z výnosov akcií

Vážené siete z matice S pre dáta výnosov akcií zo siete s parametrom $\varepsilon = 0,5$ s hranicami existencie hrany aspoň 5 zhôd, aspoň 4 zhody a aspoň 3 zhody sú zobrazené na Obr. 4.4. Z Obr. 4.4 vidíme, že ak za hranicu existencie hrany zvolíme aspoň 5 zhôd, vzniknuté zhluky sú zhlukmi obsahujúcimi prevažne akcie z rovnakého odvetvia a toto zhlukovanie je podobné zhlukovaniu po odvetviach. Po znižovaní hranice existencie hrany na aspoň 4 alebo aspoň 3 zhody, vytvorené zhluky už obsahujú viac akcií, ktoré sú z rôznych odvetví. Navyše si môžeme všimnúť, že pri sieti s hranicou existencie hrany aspoň 3 zhody sa do zhlukov dostali aj niektoré vrcholy, ktoré boli pri vyšších nárokoch na hranicu existencie hrany izolované. Modularitu pre siete s rôznymi hranicami existencie hrany sme zhrnuli v Tabuľke 4.3. Najvyššiu modularitu opäť dosiahlo zhlukovanie v sieti s aspoň 5 zhodami.

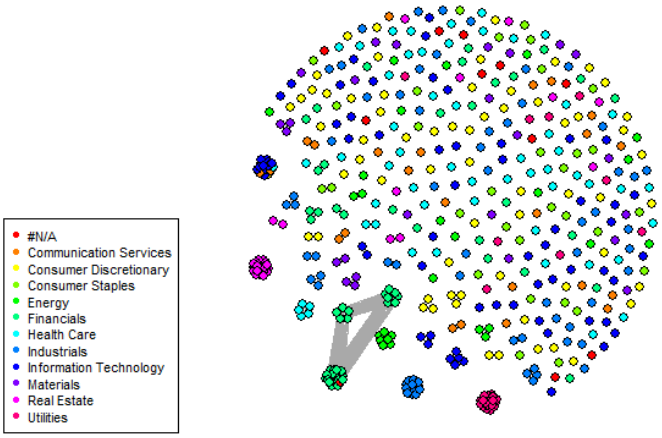
Vážené siete z matice S pre dáta výnosov akcií zo siete s parametrom $\varepsilon = 0,6$ s hranicami existencie hrany aspoň 5 zhôd, aspoň 4 zhody a aspoň 3 zhody sú zobrazené na Obr. 4.5. Z Obr. 4.5 môžeme opäť vidieť, že pri hranici existencie hrany aspoň 5 zhôd sú v zhlukoch prevažne akcie z rovnakého odvetvia. Pri znížení nárokov na hranicu existencie hrany na aspoň 4 zhody môžeme pozorovať vytvorenie niekoľkých veľkých zhlukov obsahujúcich akcie bez ohľadu na to, do akého odvetvia patria. Môžeme si všimnúť, že do týchto zhlukov boli zaradené aj vrcholy, ktoré boli pri sieti s hranicou existencie hrany aspoň 5 zhôd izolované. Hodnoty modularity pre rôzne hranice



(a) hranica existencie hrany aspoň 5 zhôd

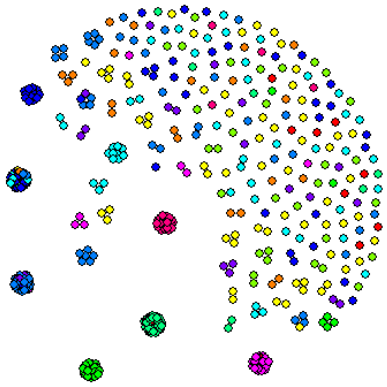


(b) hranica existencie hrany aspoň 4 zhody

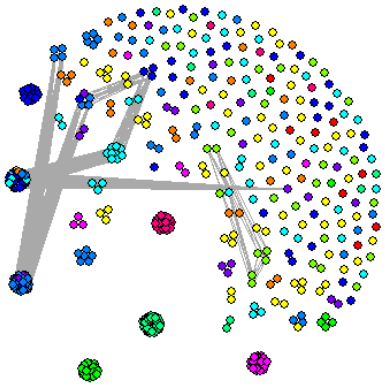


(c) hranica existencie hrany aspoň 2 zhody

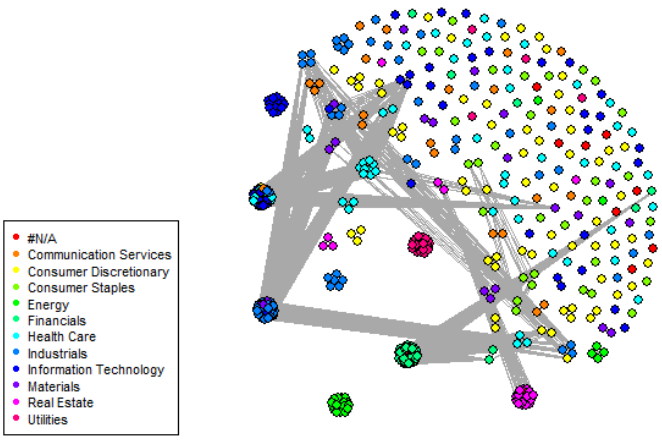
Obr. 4.3: Vážené siete z dát výnosov akcií zo siete s parametrom $\varepsilon = 0,4$



(a) hranica existencie hrany aspoň 5 zhôd



(b) hranica existencie hrany aspoň 4 zhody



(c) hranica existencie hrany aspoň 3 zhody

Obr. 4.4: Vážené siete z dát výnosov akcií zo siete s parametrom $\varepsilon = 0,5$

Hranica existencie hrany	Modularita
aspoň 5 zhôd	0,7648
aspoň 4 zhody	0,6468
aspoň 3 zhody	0,6243
aspoň 2 zhody	0,2741
aspoň 1 zhoda	0,2111
rozdelenie podľa odvetví	0,6066

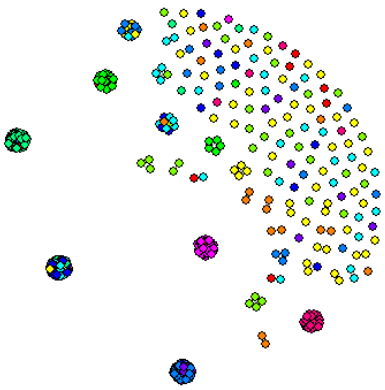
Tabuľka 4.3: Modularita na základe hranice existencie hrany vo váženej sieti a zhlukovania podľa odvetví pre sieť s parametrom $\varepsilon = 0,5$ z výnosov akcií

existencie hrany sme zhrnuli v Tabuľke 4.4, kde vidíme, že najvyššiu modularitu sme dosiahli pri zhlukovaní v sieti s aspoň 5 zhodami.

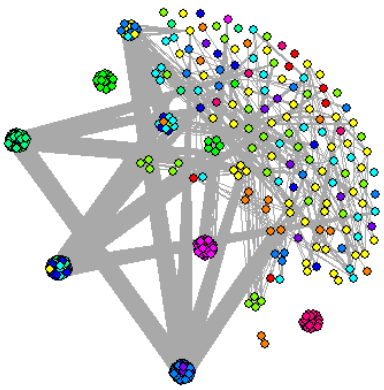
Hranica existencie hrany	Modularita
aspoň 5 zhôd	0,6513
aspoň 4 zhody	0,3445
aspoň 3 zhody	0,2290
aspoň 2 zhody	0,1217
aspoň 1 zhoda	0,0005
rozdelenie podľa odvetví	0,2327

Tabuľka 4.4: Modularita na základe hranice existencie hrany vo váženej sieti a zhlukovania podľa odvetví pre sieť s parametrom $\varepsilon = 0,6$ z výnosov akcií

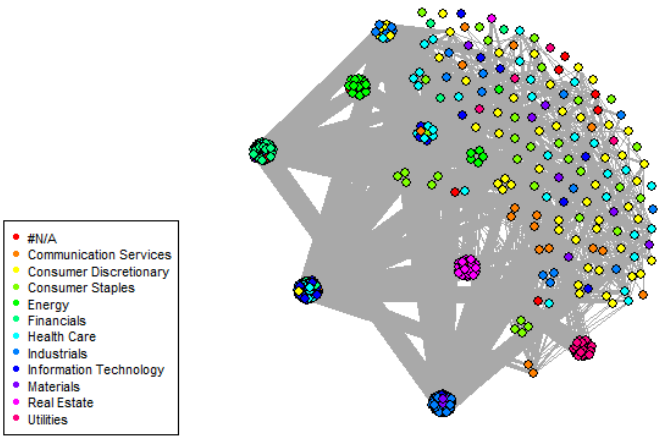
Z výsledkov vidíme, že najvyššiu hodnotu modularity zhlukovania, sme vo všetkých prípadoch (v sieťach s ε na všetkých úrovniach) dosiahli pre váženú sieť s hranicou existencie hrany aspoň 5 zhôd. Tieto siete vo všeobecnosti obsahujú veľa izolovaných vrcholov a väčší počet zhlukov v porovnaní so sieťami s nižšími nárokmi na hranicu existencie hrany. Vo všetkých prípadoch pri znižovaní hranice na existenciu hrany vidíme spájanie zhlukov do väčších zhlukov. V prípade vážených sietí vytvorených zo siete s ε na úrovni 0,4 sa do väčších zhlukov spájajú akcie, ktoré sú prevažné z rovnakého odvetvia. Pri vážených sieťach vytvorených zo sietí s ε na úrovni 0,5 a 0,6 vidíme



(a) hranica existencie hrany aspoň 5 zhôd



(b) hranica existencie hrany aspoň 4 zhody



(c) hranica existencie hrany aspoň 3 zhody

Obr. 4.5: Vážené siete z dát výnosov akcií zo siete s parametrom $\varepsilon = 0,6$

pri znižovaní nárokov na hranicu existencie hrany spájanie takých zhlukov akcií, ktoré nepatria do rovnakého odvetvia.

Pri porovnávaní modularity zhlukovania podľa odvetví s modularitou ostatných zhlukovaní v sieti s určitou hodnotou ε si môžeme všimnúť, že modularita zhlukovania po odvetviach je podobná, teda nie je výrazne horšia ako najlepšia možná modularita, najmä pre siete vytvorené s nižšou hodnotou ε. Modularita zhlukovania po odvetviach nemá šancu byť výrazne lepšia oproti modularite zhlukovaní podľa nášho algoritmu, nakoľko väčšina algoritmov zhlukovania, ktoré vstupujú do nášho algoritmu modularitu optimalizujú.

Záver

Témou diplomovej práce bolo zhlukovanie časových radov založené na algoritmoch detekcie komunít v sociálnych sieťach. Cieľom bolo vysvetliť algoritmus z článku [8], implementovať ho na testovacie a vlastné dáta použitím rôznych nastavení daných parametrov a pokúsiť sa ho „vylepšiť“ tak, že algoritmus zbehneme niekoľkokrát s rôznym nastavením parametrov a výsledky vhodne skombinujeme.

Táto diplomová práca obsahuje 4 kapitoly.

V prvej kapitole uvádzame čitateľa do problematiky a vysvetľujeme teoretické poznatky potrebné k pochopeniu algoritmu. Zadefinovali sme pojmy a urobili sme podrobný prehľad mier vzdialenosti, algoritmov vytvorenia siete a algoritmov detekcie komunít, s ktorými sme v diplomovej práci bližšie pracovali.

V druhej kapitole sme vysvetlili algoritmus navrhnutý v článku [8]. Vylepšenie algoritmu [8] spočíva vo vhodnom skombinovaní algoritmov s rôznym nastavením parametrov, počítaním zhôd týchto algoritmov a následnej klasifikácii testovacích dát. Vysvetlili sme, ako sme počítali, či sa jednotlivé algoritmy (algoritmy s rôznym nastavením parametrov) zhodli, ako aj vytvorenie váženej siete z časových radov na základe súčtu zhôd algoritmov a zostrojenie klasifikátora časových radov.

V tretej kapitole sme algoritmus aplikovali na trénovacie a testovacie dáta z UCR Time Series Classification Archive [12]. Podrobne sme vysvetlili celý proces algoritmu a klasifikácie na datasetoch *coffee* a *gunpoint*. Pri analýze dát *gunpoint* sme ukázali, že pri klasifikácii časových radov je potrebné zamerať sa na priebeh časových radov a navrhli sme vylepšenie algoritmu. Algoritmus sme aplikovali aj na ďalšie datasety z [12], výsledky zhŕňame v podkapitole 3.3.

Aplikáciu algoritmu na reálne dáta uvádzame v štvrtej kapitole, kde opisujeme proces zhlukovania časových radov z odvetvia financií. Dátami sú denné výnosy akcií indexu SP 500 za rok 2018. Vysvetľujeme proces dopočítania chýbajúcich hodnôt v dátach a navrhujeme algoritmus s vhodnými parametrami. Skúmame, ako sa menia zhluky pri jednotlivých zhlukovaniach a zároveň porovnávame výsledky našich zhlukovaní so zhlukovaním po odvetviach, z ktorých výnosy akcií pochádzajú.

Zoznam použitej literatúry

[1] Aghabozorgi S., Shirkhorshidi A. S., Wah T. Y.: *Time-series clustering – A decade review*, In Information Systems 53 (2015), p. 16-38

[2] Batista G. E., Wang, X., Keogh, E. J. *A Complexity-Invariant Distance Measure for Time Series*, In Proceedings of the 2011 SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics (2011), p. 699-710

[3] Berndt, D. J., Clifford J.: *Using Dynamic Time Warping to Find Patterns in Time Series*, In KDD workshop Vol. 10, No. 16 (1994), 359-370

[4] Blondel V. D. et al.: *Fast unfolding of communities in large networks*, In Journal of statistical mechanics: theory and experiment 2008.10 (2008), P10008

[5] Brandes U., et al..: *On Finding Graph Clusterings with Maximum Modularity*, In International Workshop on Graph-Theoretic Concepts in Computer Science. Springer, Berlin, Heidelberg (2007), p. 121-132

[6] Clauset, A., Newman M. EJ., Moore C.: *Finding community structure in very large networks*, In Physical Review E 70.6 (2004), 066111

[7] De Lucas D. C.: *Classification Techniques for Time Series and Functional Data*, PhD thesis, Universidad Carlos III de Madrid (2010)

[8] Ferreira L. N., Zhao L.: *A Time Series Clustering Technique based on Community Detection in Networks*, In Procedia Computer Science 53 (2015), p. 183-190

[9] Ferreira L. N., Zhao L.: kód v jazyku R k článku *A Time Series Clustering Technique based on Community Detection in Networks*, dostupné na internete (2.5.2020): `https://github.com/lnferreira/time_series_clustering_via_community_detection`

[10] Fretzos E., Gratsias K., Theodoridis Y.: *Index-based Most Similar Trajectory Search*, In 2007 IEEE 23rd International Conference on Data Engineering, IEEE (2007), p. 816-825

[11] Harman, R.: *Multivariate Statistical Analysis*, poznámky k vybraným prednáškam predmetu Viacrozmerné štatistické analýzy 2, učebný text (2018), dostupné na internete (19.12.2019): `http://www.iam.fmph.uniba.sk/ospm/Harman/VSAp.pdf`

[12] Hoang A. D., et al..: *The UCR Time Series Classification Archive* 2019 dostupné na internete (13.1.2020): `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`

[13] *Kronecker delta*, dostupné na internete (17.4.2020): `https://en.wikipedia.org/wiki/Kronecker_delta`

[14] Kyaagba S.: *Dynamic Time Warping with Time Series*, 2018, dostupné na internete (28.3.2020): `https://medium.com/@shachiakyaagba_41915/dynamic-time-warping-with-time-series-1f5c05fb8950`

[15] List of S&P 500 companies, dostupné na internete (10.2.2020): `https://en.wikipedia.org/wiki/List_of_S%26P_500_companies`

[16] Mantegna R. N.: *Hierarchical Structure in Financial Markets*, In The European Physical Journal B - Condensed Matter and Complex Systems 11.1 (1999), p. 193-197

[17] Montero P., et al.: *TSclust: An R Package for Time Series Clustering*, In Journal of Statistical Software, 62.1 (2014), p. 1-43

[18] Moritz S, Bartz-Beielstein T *imputeTS: Time Series Missing Value Imputation in R* In The R Journal, 9.1 (2017) p. 207-218

[19] Möller-Levet C. S., et al.: *Fuzzy clustering of short time series and unevenly distributed sampling points*, In International Symposium on Intelligent Data Analysis, Springer, Berlin, Heidelberg (2003), p. 330-340

[20] Optimal community structure - help k funkcii `cluster_optimal`, dostupné na internete (3.5.2020): `https://igraph.org/r/doc/cluster_optimal.html`

[21] Perlin, M.:*BatchGetSymbols: Downloads and Organizes Financial Data for Multiple Tickers* R package version 2.5.4., (2019)

[22] Pons P., Latapy M.: *Computing Communities in Large Networks Using Random Walks*, In International Symposium on Computer and Infomation Sciences, Springer, Berlin, Heidelberg (2005), p. 284-293

[23] Prindle, D.: *This pocket-sized molecular spectrometer tells you the chemical makeup of foods* (2014), dostupné na internete (14.1.2019): `https://www.digitaltrends.com/cool-tech/pocket-sized-molecular-spectrometer-tells-chemical-makeup-foods/`

[24] Rosvall M., Bergstrom C.T.: *Maps of random walks on complex networks reveal community structure*, In Proceedings of the National Academy of Sciences, 105.4 (2008), p. 1118-1123

[25] Shumway, R. H., Stoffer, D. S.: *Time series analysis and its applications: with R examples*, Springer (2017)

[26] Stehlíková B.: Hľadanie komunít v sieťach - časť II. - poznámky k prednáške, učebný text, dostupné na internete (17.4.2020): `http://www.iam.fmph.uniba.sk/institute/stehlikova/siete20/siete04_poznamky.pdf`

[27] Tangborn A.: *Wavelet Transforms in Time Series Analysis, Global Modeling and Assimilation Office*, dostupné na internete (5.4.2020): `https://www2.atmos.umd.edu/~ekalnay/syllabi/AOSC630/Wavelets_2010.pdf`

[28] Tsiporkova, E. : *Dynamic Time Warping*, dostupné na internete (28.3.2020): `http://www.mathcs.emory.edu/~lxiong/cs730_s13/share/slides/searching_sigkdd2012_DTW.pdf`

[29] Raghavan U. N., Réka A., Soundar K.: *Near linear time algorithm to detect community structures in large-scale networks*, In Physical Review E 76.3 (2007), 036106

[30] Zhang, H.,et al.: *Unsupervised feature extraction for time series clustering using orthogonal wavelet transform*, In Informatica 30.3 (2006)

Príloha

```
library(TSdist)
library(TSclust)
library(scales)
library(igraph)
set.seed(123)

################POTREBNE FUNKCIE################
#znormalizovanie matice vzdialenosti, zdroj: [9]
dist.normalize <- function(dist) {
  distN <- as.matrix(dist)
  distNorm <- matrix(0, nrow(dist), ncol(dist))
  d <- distN[upper.tri(distN)]
  d <- rescale(d)
  distNorm[upper.tri(distNorm)] <- d
  distNorm <- distNorm + t(distNorm)
  distNorm
}

#funkcie na vytvorenie siete, zdroj: [9]
net.knn.create <- function(dist, k) {
  d <- as.matrix(dist) + diag(Inf, nrow(dist), ncol(dist))
  net <- graph.empty(nrow(dist), directed = FALSE)
  knnRows <- apply(d, MARGIN = 1, function(x) which(x %in% sort(x,
      method="quick")[1:k])[1:k])
  knnRows <- matrix(knnRows, nrow=k)
  for (i in 1:nrow(knnRows)) {
    for (j in 1:ncol(knnRows)) {
      if (!are.connected(net, j, knnRows[i,j])) {
        net <- add.edges(net, c(j, knnRows[i,j]))}}}
  net
}

net.epsilon.create <- function(dist, epsilon) {
  n = matrix(0, ncol(dist), nrow(dist))
  n[dist <= epsilon] = 1;
  return(graph.adjacency(n, mode="undirected", diag=FALSE));
```

```
}

#funkcia na vytvorenie matice S
f <- function(i,j) {
  1 * (zhluky$membership[i] ==  zhluky$membership[j])
}
```

Listing 1: Funkcie potrebné k algoritmu

```
#################1.SIET################
#nacitanie dat
data_full <- read.table("gunpoint_TRAIN.txt", header=FALSE, sep="\t")
labels <- data_full[ , 1]
data <- data_full[ , 2:ncol(data_full)]
data_test <- read.table("gunpoint_TEST.txt", header=FALSE, sep = "\t")
matplot(t(data), type="l", lty = 1, col = labels)

#vypocet vzdialenosti
tsdiss <- TSDatabaseDistances(data, Y = NULL, distance = "dtw") #dtw
tsdiss.norm <- dist.normalize(tsdiss)
rownames(tsdiss.norm) <- labels
colnames(tsdiss.norm) <- labels

N <- length(labels)
matrix_all <- matrix(0, nrow=N, ncol=N)
rownames(matrix_all) <- labels
colnames(matrix_all) <-labels

for (i in 12:12){
  net <- net.knn.create(tsdiss.norm, i)
  V(net)$name <- labels
  walktrap <- cluster_walktrap(net)
  fastgreedy <- cluster_fast_greedy(net)
  multilevel <- cluster_louvain(net)
  infomap <- cluster_infomap(net)
  labelprop <- cluster_label_prop(net)

```

```
  zhluky <- walktrap
  walktrap_m <- outer(1:N, 1:N, f)
  zhluky <- fastgreedy
  fastgreedy_m <- outer(1:N, 1:N, f)
  zhluky <- multilevel
  multilevel_m <- outer(1:N, 1:N, f)
  zhluky <- infomap
  infomap_m <- outer(1:N, 1:N, f)
  zhluky <- labelprop
  labelprop_m <- outer(1:N, 1:N, f)

  matrix_all <- matrix_all + walktrap_m + fastgreedy_m + multilevel_m
     + infomap_m + labelprop_m
}

g4 <- graph_from_adjacency_matrix(matrix_all>=4, weighted = TRUE, diag
    = FALSE, mode = "undirected")
plot(g4, edge.width=matrix_all/2, vertex.color=labels)
comp <- components(g4)
comp #data skupiny 1 su v komponente 3, data skupiny 2 su v komponente
    2 (v klasifikatore usporiadane v spravnom poradi)

################KLASIFIKATOR 1#################
klasifikacia <- matrix(0, nrow=nrow(data_test), ncol=5)
klasifikacia[ , 1] <- as.numeric(data_test[ , 1])

for(k in 1:nrow(data_test)) {
  data <- read.table("gunpoint_TRAIN.txt", header=FALSE, sep = "\t")
  labels <- data[ , 1]
  data <- data[ , 2:ncol(data)]
  data_test <- read.table("gunpoint_TEST.txt", header=FALSE, sep = "\t
     ")
  data_df <- rbind(data, data_test[k,-1])  # 2. casovy rad
  data <- data_df
  N <- nrow(data)
  matrix_all <- matrix(0, nrow=N, ncol=N)
  rownames(matrix_all) <- c(labels,"X")
  colnames(matrix_all) <- c(labels,"X")
```

```
104
105 tsdiss <- TSDatabaseDistances(data, Y = NULL, distance = "dtw") #dtw
106 tsdiss.norm <- dist.normalize(tsdiss)
107
108 for (i in 12:12) {
109   net <- net.knn.create(tsdiss.norm, i)
110   walktrap <- cluster_walktrap(net)
111   fastgreedy <- cluster_fast_greedy(net)
112   multilevel <- cluster_louvain(net)
113   infomap <- cluster_infomap(net)
114   labelprop <- cluster_label_prop(net)
115
116   zhluky <- walktrap
117   walktrap_m <- outer(1:N, 1:N, f)
118   zhluky <- fastgreedy
119   fastgreedy_m <- outer(1:N, 1:N, f)
120   zhluky <- multilevel
121   multilevel_m <- outer(1:N, 1:N, f)
122   zhluky <- infomap
123   infomap_m <- outer(1:N, 1:N, f)
124   zhluky <- labelprop
125   labelprop_m <- outer(1:N, 1:N, f)
126
127   matrix_all <- matrix_all + walktrap_m + fastgreedy_m + multilevel_
        m + infomap_m + labelprop_m
128 }
129
130 g <- graph_from_adjacency_matrix(matrix_all >= 4, weighted = TRUE,
     diag = FALSE, mode = "undirected")
131 com1 <- make_clusters(g, membership = c(comp$membership,1))
132 mod1 <- modularity(com1)
133 mod1
134 com2 <- make_clusters(g, membership = c(comp$membership,2))
135 mod2 <- modularity(com2)
136 mod2
137 com3 <- make_clusters(g, membership = c(comp$membership,3))
138 mod3 <- modularity(com3)
139 mod3
```

```
140
141   klasifikacia[k, 2] <- which.max(c(modularity(com3), modularity(com2)
          , modularity(com1))) #spravne usporiadane
142   klasifikacia[k, 3:5] <- c(mod3, mod2, mod1)
143   klasifikacia
144 }
145 klasifikacia
146
147 trieda1 <- klasifikacia[klasifikacia[ , 2]== 1, ]
148 nrow(trieda1[trieda1[ , 1]==1, ]) #spravne zaradene cR z triedy 1
149 nrow(trieda1) #celkovy pocet zaradeni do triedy 1
150 trieda2 <- klasifikacia[klasifikacia[ , 2]== 2, ]
151 nrow(trieda2[trieda2[ , 1]==2, ]) #spravne zaradene CR z triedy 2
152 nrow(trieda2) #celkovy pocet zaradeni do triedy 2
153 trieda3 <- klasifikacia[klasifikacia[ , 2]== 3, ]
154 nrow(trieda3)
155
156 #vyber testovacich dat zaradenych do zmiesaneho zhluku
157 data2_2test <- data_test[klasifikacia[ , 2]==3, ]
158 write.csv(data2_2test, file="gunpoint2_test.txt")
159
160 ################2.SIET################
161 data_comp <- cbind(comp$membership, data_full)
162 data_comp_df <- data_comp[data_comp$`comp$membership`== 1, ]
163 labels2 <- data_comp_df[ , 2]
164 data2_2 <- data_comp_df[ , 2:ncol(data_comp_df)]
165 write.csv(data2_2, file="gunpoint2_train.txt")
166 data2 <- data_comp_df[ , 3:ncol(data_comp_df)]
167 #vsetky data z 1.klastru
168 matplot(t(data2), type="l", lty = 1, col = labels2)
169
170 #rozdelenie trenovacich dat na testovaciu a validacnu cast
171 split <- sample(1:2, size=nrow(data2), prob=c(0.7,0.3), replace = TRUE
      )
172 train.data2 <- data2[split==1, ]
173 valid.data2 <- data2[split==2, ]
174 train.labels2 <- labels2[split==1]
175 valid.labels2 <- labels2[split==2]
```

```
train.data2_full <- cbind(train.labels2, train.data2)
write.csv(train.data2_full, file="traindata2.txt")
valid.data2_full <- cbind(valid.labels2, valid.data2)
write.csv(valid.data2_full, file="validdata2.txt")

#nacitanie trenovacej a validacnej casti ##
train.data2 <- read.table("traindata2.txt", header=TRUE, sep=",")
train.data2 <- train.data2[ , 2:ncol(train.data2)]
train.labels2 <- train.data2[ , 1]
train.data2 <- train.data2[ , 2:ncol(train.data2)]
valid.data2 <- read.table("validdata2.txt", header=TRUE, sep=",")
valid.data2 <- valid.data2[ , 2:ncol(valid.data2)]
valid.labels2 <- valid.data2[ , 1]
valid.data2 <- valid.data2[ , 2:ncol(valid.data2)]

################TESTOVANIE CASOVYCH USEKOV################
##prvych 40 hodnot
train.data2p <- train.data2[ , 1:40]
#vypocet vzdialenosti
tsdiss2 <- TSDatabaseDistances(train.data2p, Y = NULL, distance = "dtw
    ") #dtw
tsdiss.norm2 <- dist.normalize(tsdiss2)
rownames(tsdiss.norm2) <- train.labels2
colnames(tsdiss.norm2) <- train.labels2
N2 <- length(train.labels2)
matrix_all2 <- matrix(0, nrow=N2, ncol=N2)
rownames(matrix_all2) <- train.labels2
colnames(matrix_all2) <-train.labels2

for (i in 3:3) {
  net2 <- net.knn.create(tsdiss.norm2, i)
  walktrap2 <- cluster_walktrap(net2)
  fastgreedy2 <- cluster_fast_greedy(net2)
  multilevel2 <- cluster_louvain(net2)
  infomap2 <- cluster_infomap(net2)
  labelprop2 <- cluster_label_prop(net2)

  zhluky <- walktrap2
```

```
  walktrap_m2 <- outer(1:N2, 1:N2, f)
  zhluky <- fastgreedy2
  fastgreedy_m2 <- outer(1:N2, 1:N2, f)
  zhluky <- multilevel2
  multilevel_m2 <- outer(1:N2, 1:N2, f)
  zhluky <- infomap2
  infomap_m2 <- outer(1:N2, 1:N2, f)
  zhluky <- labelprop2
  labelprop_m2 <- outer(1:N2, 1:N2, f)

  matrix_all2 <- matrix_all2 + walktrap_m2 + fastgreedy_m2 +
     multilevel_m2 + infomap_m2 + labelprop_m2
}

g2_5 <- graph_from_adjacency_matrix(matrix_all2>=5, weighted = TRUE,
    diag = FALSE, mode = "undirected")
plot(g2_5, edge.width=matrix_all/2, vertex.color=train.labels2, vertex
    .label=NA)
comp2 <- components(g2_5)

data_validacia <- read.table("validdata2.txt", header=TRUE, sep=",")
data_validacia <- data_validacia[ , 2:ncol(data_validacia)]
klasifikacia2_valid <- matrix(0, nrow=nrow(data_validacia), ncol=4)
klasifikacia2_valid[ , 1] <- as.numeric(data_validacia[ , 1])
data_validacia <- data_validacia[ , 2:41]

for(k in 1:nrow(data_validacia)){
  data2 <- read.table("traindata2.txt", header=TRUE, sep = ",")
  data2 <- data2[ , 2:ncol(data2)]
  labels <- as.numeric(data2[ , 1])
  data2 <- data2[ , 2:41]
  data_test22 <- read.table("validdata2.txt", header=TRUE, sep = ",")
  data_test22 <- data_test22[ , 3:42]
  data_df2 <- rbind(data2, data_test22[k, ])  # 2. casovy rad
  data <- data_df2
  N <- nrow(data)
  matrix_all <- matrix(0, nrow=N, ncol=N)
  rownames(matrix_all) <- c(labels,"X")
```

```
248   colnames(matrix_all) <- c(labels,"X")
249
250   tsdiss <- TSDatabaseDistances(data, Y = NULL, distance = "dtw") #dtw
251   tsdiss.norm <- dist.normalize(tsdiss)
252
253   for (i in 3:3){
254     net <- net.knn.create(tsdiss.norm, i)
255     walktrap <- cluster_walktrap(net)
256     fastgreedy <- cluster_fast_greedy(net)
257     multilevel <- cluster_louvain(net)
258     infomap <- cluster_infomap(net)
259     labelprop <- cluster_label_prop(net)
260
261     zhluky <- walktrap
262     walktrap_m <- outer(1:N, 1:N, f)
263     zhluky <- fastgreedy
264     fastgreedy_m <- outer(1:N, 1:N, f)
265     zhluky <- multilevel
266     multilevel_m <- outer(1:N, 1:N, f)
267     zhluky <- infomap
268     infomap_m <- outer(1:N, 1:N, f)
269     zhluky <- labelprop
270     labelprop_m <- outer(1:N, 1:N, f)
271
272     matrix_all <- matrix_all + walktrap_m + fastgreedy_m + multilevel_
          m + infomap_m + labelprop_m
273     }
274   g <- graph_from_adjacency_matrix(matrix_all>=5, weighted = TRUE,
       diag = FALSE, mode = "undirected")
275
276   com1 <- make_clusters(g, membership = c(comp2$membership,1))
277   mod1 <- modularity(com1)
278   mod1
279   com2 <- make_clusters(g, membership = c(comp2$membership,2))
280   mod2 <- modularity(com2)
281   mod2
282
283   klasifikacia2_valid[k, 2] <- which.max(c(modularity(com2),
```

```
      modularity(com1)))
  klasifikacia2_valid[k, 3:4] <- c(mod2, mod1)
  klasifikacia2_valid
}
klasifikacia2_valid

length(which(klasifikacia2_valid[ , 1] == klasifikacia2_valid[ , 2]))/
    nrow(klasifikacia2_valid) #v kolkych pripadoch sa trafil

##stredna cast dat
train.data2p <- train.data2[ , 41:110]
#vypocet vzdialenosti
tsdiss2 <- TSDatabaseDistances(train.data2p, Y = NULL, distance = "dtw
    ") #dtw
tsdiss.norm2 <- dist.normalize(tsdiss2)
rownames(tsdiss.norm2) <- train.labels2
colnames(tsdiss.norm2) <- train.labels2

N2 <- length(train.labels2)
matrix_all2 <- matrix(0, nrow=N2, ncol=N2)
rownames(matrix_all2) <- train.labels2
colnames(matrix_all2) <-train.labels2

for (i in 3:3) {
  net2 <- net.knn.create(tsdiss.norm2, i)
  walktrap2 <- cluster_walktrap(net2)
  fastgreedy2 <- cluster_fast_greedy(net2)
  multilevel2 <- cluster_louvain(net2)
  infomap2 <- cluster_infomap(net2)
  labelprop2 <- cluster_label_prop(net2)

  zhluky <- walktrap2
  walktrap_m2 <- outer(1:N2, 1:N2, f)
  zhluky <- fastgreedy2
  fastgreedy_m2 <- outer(1:N2, 1:N2, f)
  zhluky <- multilevel2
  multilevel_m2 <- outer(1:N2, 1:N2, f)
  zhluky <- infomap2
```

```
  infomap_m2 <- outer(1:N2, 1:N2, f)
  zhluky <- labelprop2
  labelprop_m2 <- outer(1:N2, 1:N2, f)

  matrix_all2 <- matrix_all2 + walktrap_m2 + fastgreedy_m2 +
     multilevel_m2 + infomap_m2 + labelprop_m2
}

g2_5 <- graph_from_adjacency_matrix(matrix_all2>=5, weighted = TRUE,
    diag = FALSE, mode = "undirected")
plot(g2_5, edge.width=matrix_all/2, vertex.color=train.labels2, vertex
    .label=NA)
comp2 <- components(g2_5)

data_validacia <- read.table("validdata2.txt", header=TRUE, sep=",")
data_validacia <- data_validacia[ , 2:ncol(data_validacia)]
klasifikacia2_valid <- matrix(0, nrow=nrow(data_validacia), ncol=4)
klasifikacia2_valid[ , 1] <- as.numeric(data_validacia[ , 1])
data_validacia <- data_validacia[ , 42:111]

for(k in 1:nrow(data_validacia)) {
  data2 <- read.table("traindata2.txt", header=TRUE, sep = ",")
  data2 <- data2[ , 2:ncol(data2)]
  labels <- as.numeric(data2[ , 1])
  data2 <- data2[ , 42:111]
  data_test22 <- read.table("validdata2.txt", header=TRUE, sep = ",")
  data_test22 <- data_test22[ , 43:112]
  data_df2 <- rbind(data2, data_test22[k, ])  # 2. casovy rad
  data <- data_df2
  N <- nrow(data)
  matrix_all <- matrix(0, nrow=N, ncol=N)
  rownames(matrix_all) <- c(labels,"X")
  colnames(matrix_all) <- c(labels,"X")

  tsdiss <- TSDatabaseDistances(data, Y = NULL, distance = "dtw") #dtw
  tsdiss.norm <- dist.normalize(tsdiss)

  for (i in 3:3) {
```

```
354     net <- net.knn.create(tsdiss.norm, i)
355     walktrap <- cluster_walktrap(net)
356     fastgreedy <- cluster_fast_greedy(net)
357     multilevel <- cluster_louvain(net)
358     infomap <- cluster_infomap(net)
359     labelprop <- cluster_label_prop(net)
360 
361     zhluky <- walktrap
362     walktrap_m <- outer(1:N, 1:N, f)
363     zhluky <- fastgreedy
364     fastgreedy_m <- outer(1:N, 1:N, f)
365     zhluky <- multilevel
366     multilevel_m <- outer(1:N, 1:N, f)
367     zhluky <- infomap
368     infomap_m <- outer(1:N, 1:N, f)
369     zhluky <- labelprop
370     labelprop_m <- outer(1:N, 1:N, f)
371 
372     matrix_all <- matrix_all + walktrap_m + fastgreedy_m + multilevel_
          m + infomap_m + labelprop_m
373   }
374   g <- graph_from_adjacency_matrix(matrix_all>=5, weighted = TRUE,
        diag = FALSE, mode = "undirected")
375 
376   com1 <- make_clusters(g, membership = c(comp2$membership,1))
377   mod1 <- modularity(com1)
378   mod1
379   com2 <- make_clusters(g, membership = c(comp2$membership,2))
380   mod2 <- modularity(com2)
381   mod2
382 
383   klasifikacia2_valid[k, 2] <- which.max(c(modularity(com2),
        modularity(com1)))
384   klasifikacia2_valid[k, 3:4] <- c(mod2, mod1)
385   klasifikacia2_valid
386 }
387 klasifikacia2_valid
388 
```

```
389 length(which(klasifikacia2_valid[ , 1] == klasifikacia2_valid[ , 2]))/
        nrow(klasifikacia2_valid) #v kolkych pripadoch sa trafil
390
391 ##poslednych 40 hodnot
392 train.data2p <- train.data2[ , 111:150]
393 #vypocet vzdialenosti
394 tsdiss2 <- TSDatabaseDistances(train.data2p, Y = NULL, distance = "dtw
        ") #dtw
395 tsdiss.norm2 <- dist.normalize(tsdiss2)
396 rownames(tsdiss.norm2) <- train.labels2
397 colnames(tsdiss.norm2) <- train.labels2
398 N2 <- length(train.labels2)
399 matrix_all2 <- matrix(0, nrow=N2, ncol=N2)
400 rownames(matrix_all2) <- train.labels2
401 colnames(matrix_all2) <-train.labels2
402
403 for (i in 3:3) {
404   net2 <- net.knn.create(tsdiss.norm2, i)
405   walktrap2 <- cluster_walktrap(net2)
406   fastgreedy2 <- cluster_fast_greedy(net2)
407   multilevel2 <- cluster_louvain(net2)
408   infomap2 <- cluster_infomap(net2)
409   labelprop2 <- cluster_label_prop(net2)
410
411   zhluky <- walktrap2
412   walktrap_m2 <- outer(1:N2, 1:N2, f)
413   zhluky <- fastgreedy2
414   fastgreedy_m2 <- outer(1:N2, 1:N2, f)
415   zhluky <- multilevel2
416   multilevel_m2 <- outer(1:N2, 1:N2, f)
417   zhluky <- infomap2
418   infomap_m2 <- outer(1:N2, 1:N2, f)
419   zhluky <- labelprop2
420   labelprop_m2 <- outer(1:N2, 1:N2, f)
421
422   matrix_all2 <- matrix_all2 + walktrap_m2 + fastgreedy_m2 +
        multilevel_m2 + infomap_m2 + labelprop_m2
423 }
```

```
424
425 g2_4 <- graph_from_adjacency_matrix(matrix_all2>=4, weighted = TRUE,
      diag = FALSE, mode = "undirected")
426 plot(g2_4, edge.width=matrix_all/2, vertex.color=train.labels2, vertex
      .label=NA)
427 comp2 <- components(g2_4)
428 comp2
429
430 data_validacia <- read.table("validdata2.txt", header=TRUE, sep=",")
431 data_validacia <- data_validacia[ , 2:ncol(data_validacia)]
432 klasifikacia2_valid <- matrix(0, nrow=nrow(data_validacia), ncol=4)
433 klasifikacia2_valid[ , 1] <- as.numeric(data_validacia[ , 1])
434 data_validacia <- data_validacia[ , 112:151]
435
436 for(k in 1:nrow(data_validacia)) {
437   data2 <- read.table("traindata2.txt", header=TRUE, sep = ",")
438   data2 <- data2[ , 2:ncol(data2)]
439   labels <- as.numeric(data2[ , 1])
440   data2 <- data2[ , 112:151]
441   data_test22 <- read.table("validdata2.txt", header=TRUE, sep = ",")
442   data_test22 <- data_test22[ , 113:152]
443   data_df2 <- rbind(data2, data_test22[k, ])  # 2. casovy rad
444   data <- data_df2
445   N <- nrow(data)
446   matrix_all <- matrix(0, nrow=N, ncol=N)
447   rownames(matrix_all) <- c(labels,"X")
448   colnames(matrix_all) <- c(labels,"X")
449
450   tsdiss <- TSDatabaseDistances(data, Y = NULL, distance = "dtw") #dtw
451   tsdiss.norm <- dist.normalize(tsdiss)
452
453   for (i in 3:3) {
454     net <- net.knn.create(tsdiss.norm, i)
455     walktrap <- cluster_walktrap(net)
456     fastgreedy <- cluster_fast_greedy(net)
457     multilevel <- cluster_louvain(net)
458     infomap <- cluster_infomap(net)
459     labelprop <- cluster_label_prop(net)
```

```
460
461     zhluky <- walktrap
462     walktrap_m <- outer(1:N, 1:N, f)
463     zhluky <- fastgreedy
464     fastgreedy_m <- outer(1:N, 1:N, f)
465     zhluky <- multilevel
466     multilevel_m <- outer(1:N, 1:N, f)
467     zhluky <- infomap
468     infomap_m <- outer(1:N, 1:N, f)
469     zhluky <- labelprop
470     labelprop_m <- outer(1:N, 1:N, f)
471
472     matrix_all <- matrix_all + walktrap_m + fastgreedy_m + multilevel_
          m + infomap_m + labelprop_m
473   }
474   g <- graph_from_adjacency_matrix(matrix_all>=4, weighted = TRUE,
        diag = FALSE, mode = "undirected")
475
476   com1 <- make_clusters(g, membership = c(comp2$membership,1))
477   mod1 <- modularity(com1)
478   mod1
479   com2 <- make_clusters(g, membership = c(comp2$membership,2))
480   mod2 <- modularity(com2)
481   mod2
482
483   klasifikacia2_valid[k, 2] <- which.max(c(modularity(com2),
        modularity(com1)))
484   klasifikacia2_valid[k, 3:4] <- c(mod2, mod1)
485   klasifikacia2_valid
486 }
487 klasifikacia2_valid
488
489 length(which(klasifikacia2_valid[ , 1] == klasifikacia2_valid[ , 2]))/
      nrow(klasifikacia2_valid) #v kolkych pripadoch sa trafil
490
491 ################KLASIFIKATOR 2###############
492 data_test2 <- read.table("gunpoint2_test.txt", header=TRUE, sep = ",")
493 data_test2 <- data_test2[ , 2:ncol(data_test2)]
```

```
klasifikacia2 <- matrix(0, nrow=nrow(data_test2), ncol=4)
klasifikacia2[ , 1] <- as.numeric(data_test2[ , 1])
data_test2 <- data_test2[ , 112:ncol(data_test2)]

for(k in 1:nrow(data_test2)) {
  data2 <- read.table("traindata2.txt", header=TRUE, sep = ",")
  data2 <- data2[ , 2:ncol(data2)]
  labels <- as.numeric(data2[ , 1])
  data2 <- data2[ , 112:ncol(data2)]
  data_test22 <- read.table("gunpoint2_test.txt", header=TRUE, sep = "
      ,")
  data_test22 <- data_test22[ , 113:ncol(data_test22)]
  data_df2 <- rbind(data2, data_test22[k, ])  # 2. casovy rad
  data <- data_df2
  N <- nrow(data)
  matrix_all <- matrix(0, nrow=N, ncol=N)
  rownames(matrix_all) <- c(labels,"X")
  colnames(matrix_all) <- c(labels,"X")

  tsdiss <- TSDatabaseDistances(data, Y = NULL, distance = "dtw") #dtw
  tsdiss.norm <- dist.normalize(tsdiss)

  for (i in 3:3) {
    net <- net.knn.create(tsdiss.norm, i)
    walktrap <- cluster_walktrap(net)
    fastgreedy <- cluster_fast_greedy(net)
    multilevel <- cluster_louvain(net)
    infomap <- cluster_infomap(net)
    labelprop <- cluster_label_prop(net)

    zhluky <- walktrap
    walktrap_m <- outer(1:N, 1:N, f)
    zhluky <- fastgreedy
    fastgreedy_m <- outer(1:N, 1:N, f)
    zhluky <- multilevel
    multilevel_m <- outer(1:N, 1:N, f)
    zhluky <- infomap
    infomap_m <- outer(1:N, 1:N, f)
```

```
531     zhluky <- labelprop
532     labelprop_m <- outer(1:N, 1:N, f)
533 
534     matrix_all <- matrix_all + walktrap_m + fastgreedy_m + multilevel_
            m + infomap_m + labelprop_m
535   }
536   g <- graph_from_adjacency_matrix(matrix_all>=4, weighted = TRUE,
          diag = FALSE, mode = "undirected")
537 
538   com1 <- make_clusters(g, membership = c(comp2$membership,1))
539   mod1 <- modularity(com1)
540   mod1
541   com2 <- make_clusters(g, membership = c(comp2$membership,2))
542   mod2 <- modularity(com2)
543   mod2
544 
545   klasifikacia2[k, 2] <- which.max(c(modularity(com2), modularity(com1
          )))
546   klasifikacia2[k, 3:4] <- c(mod2, mod1)
547   klasifikacia2
548 }
549 klasifikacia2
550 
551 trieda1_2 <- klasifikacia2[klasifikacia2[ , 2]== 1, ]
552 nrow(trieda1_2[trieda1_2[ , 1]== 1, ]) #spravne zaradene CR z triedy 1
553 nrow(trieda1_2)
554 trieda2_2 <- klasifikacia2[klasifikacia2[ , 2]== 2, ]
555 nrow(trieda2_2[trieda2_2[ , 1]== 2, ]) #spravne zaradene CR z triedy 2
556 nrow(trieda2_2)
557 
558 #celkovy pocet spravne zaradenych testovacich CR
559 nrow(trieda1[trieda1[ , 1]==1, ]) + nrow(trieda2[trieda2[ , 1]==2, ])
        + nrow(trieda1_2[trieda1_2[ , 1]== 1, ]) +
560   nrow(trieda2_2[trieda2_2[ , 1]== 2, ])
561 #celkovy pocet testovacich CR
562 nrow(data_test)
```

Listing 2: Aplikácia algoritmu na dáta *gunpoint*

```r
library(BatchGetSymbols)

first.date <- as.Date("2018-01-01")
last.date <- as.Date("2018-12-31")
df.SP500 <- GetSP500Stocks()
tickers <- df.SP500$company
l.out <- BatchGetSymbols(tickers = tickers, first.date = first.date,
    last.date = last.date)

#vyber nazvu akcie, datumu a adjusted price
data <- l.out$df.tickers[ , 6:8]
#uchovanie toho, ci sa ceny akcie dotiahli cele
decision <- l.out$df.control[ , c(1,6)]
count(decision[decision$threshold.decision == "KEEP", ])
count(decision[decision$threshold.decision == "OUT", ])

attach(data)
#uprava formatu dat - aby boli jednotlive casove rady v riadkoch
data <- reshape(data, idvar = "ticker", timevar = "ref.date",
    direction = "wide")
#uz by sme mali mat len data s informaciou KEEP
#pridanie stlca s informaciou o tom, ci si mam ponechat data alebo nie
data <- merge(x = data, y = decision, by = "ticker", all.x = TRUE)
#skontrolovanie toho, ci mame naozaj len data s informaciou KEEP
count(data[data$threshold.decision == "KEEP", ])
count(data[data$threshold.decision == "OUT", ])

attr(data, "row.names") <- data[, 1]

#zachovanie len casovych radov
data <- data[, 2:ncol(data)-1]
write.csv(data, file = "cenySP500.csv")
detach(data)

#nacitanie cien akcii
data <- read.csv("cenySP500.csv", header = TRUE, sep = ",",
    stringsAsFactors = FALSE)
```

```
attr(data, "row.names") <- data[, 1]
data <- data[, 2:ncol(data)]

#zistenie, v ktorych riadkoch sa nachadzaju v datach NA
for (i in 1:nrow(data)){
  if (any(is.na(data[i, ]))){
    print(i)
  }
}
#130, 288, 370

library(imputeTS)
for (i in 1:nrow(data)){
  if (any(is.na(data[i, ]))){
    data[i, ] <- t(na_kalman(t(data[i, ]), model = "auto.arima"))
  }
}

#vypocet logaritmickych vynosov
vynosy <- apply(log(data[, 1:ncol(data)]), MARGIN = 1, FUN = diff)
vynosy <- t(vynosy)

#nacitanie vektora odvetvi
industries <- read.csv("industries.csv", header=TRUE, sep=";")

#vypocet vzdialenosti a aplikovanie algoritmu
tsdiss <- sqrt(2-2*cor(t(vynosy)))
tsdiss.norm <- dist.normalize(tsdiss)
rownames(tsdiss.norm) <- labels
colnames(tsdiss.norm) <- labels

N <- length(labels)
matrix_all <- matrix(0, nrow=N, ncol=N)
rownames(matrix_all) <- labels
colnames(matrix_all) <-labels

net <- net.epsilon.create(tsdiss.norm, 0.4)
#vypocet modularity v pripade, ak by bola siet rozdelena podla odvetvi
```

```
635 com <- make_clusters(net, membership=industries[ , 3])
636 mod <- modularity(com)
637
638 walktrap <- cluster_walktrap(net)
639 fastgreedy <- cluster_fast_greedy(net)
640 multilevel <- cluster_louvain(net)
641 infomap <- cluster_infomap(net)
642 labelprop <- cluster_label_prop(net)
643
644 zhluky <- walktrap
645 walktrap_m <- outer(1:N, 1:N, f)
646 zhluky <- fastgreedy
647 fastgreedy_m <- outer(1:N, 1:N, f)
648 zhluky <- multilevel
649 multilevel_m <- outer(1:N, 1:N, f)
650 zhluky <- infomap
651 infomap_m <- outer(1:N, 1:N, f)
652 zhluky <- labelprop
653 labelprop_m <- outer(1:N, 1:N, f)
654 matrix_all <- matrix_all + walktrap_m + fastgreedy_m + multilevel_m +
      infomap_m + labelprop_m
655
656 #vytvorenie vazenych sieti a vypocet modularity, v pripade, ze
      komponenty sieti budu zhluky
657 g <- graph_from_adjacency_matrix(matrix_all>=5, weighted = TRUE, diag
      = FALSE, mode = "undirected")
658 plot.igraph(g, edge.width=matrix_all/2, vertex.size=5, vertex.label=NA
      , vertex.color=rainbow(12)[industries[,3]])
659 comp <- components(g)
660 modularity(make_clusters(g, membership=comp$membership))
661
662 g <- graph_from_adjacency_matrix(matrix_all>=4, weighted = TRUE, diag
      = FALSE, mode = "undirected")
663 plot.igraph(g, edge.width=matrix_all/2, vertex.size=5, vertex.label=NA
      , vertex.color=rainbow(12)[industries[,3]])
664 comp <- components(g)
665 modularity(make_clusters(g, membership=comp$membership))
666
```

```
667 g <- graph_from_adjacency_matrix(matrix_all>=3, weighted = TRUE, diag
    = FALSE, mode = "undirected")
668 plot.igraph(g, edge.width=matrix_all/2, vertex.size=5, vertex.label=NA
    , vertex.color=rainbow(12)[industries[,3]])
669 comp <- components(g)
670 modularity(make_clusters(g, membership=comp$membership))
671
672 g <- graph_from_adjacency_matrix(matrix_all>=2, weighted = TRUE, diag
    = FALSE, mode = "undirected")
673 plot.igraph(g, edge.width=matrix_all/2, vertex.size=5, vertex.label=NA
    , vertex.color=rainbow(12)[industries[,3]])
674 comp <- components(g)
675 modularity(make_clusters(g, membership=comp$membership))
676
677 g <- graph_from_adjacency_matrix(matrix_all>=1, weighted = TRUE, diag
    = FALSE, mode = "undirected")
678 plot.igraph(g, edge.width=matrix_all/2, vertex.size=5, vertex.label=NA
    , vertex.color=rainbow(12)[industries[,3]])
679 comp <- components(g)
680 modularity(make_clusters(g, membership=comp$membership))
```

Listing 3: Príprava dát - výnosov akcií (stiahnutie a dopočítanie chýbajúcich hodnôt), porovnanie s odvetviami