VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

SYSTÉM PRO DOPORUČOVÁNÍ FILMŮ

MOVIE RECOMMENDER SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PAVEL JANKO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MARTIN ŠŮSTEK**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Janko Pavel**
Program: Informační technologie
Název: **Systém pro doporučování filmů**
Movie Recommender System
Kategorie: Umělá inteligence

Zadání:

1. Prostudujte problematiku doporučovacích systémů, zaměřte se na metody založené na kolaborativním filtrování.
2. Navrhněte doporučovací systém využívající jednu či více dostupných datových sad, které jsou vhodné pro vyhodnocení kvality doporučení dle standardních evaluačních metrik.
3. Navržený systém implementujte.
4. Navrhněte a proveďte minimálně dva experimenty s aktuálním systémem a sledujte, jaký mají vliv na kvalitu doporučení dle použitých metrik.
5. Zhodnoťte dosažené výsledky a porovnejte s již existujícími řešeními pomocí zvolených metrik.

Literatura:

- F. Ricci, L. Rokach, B. Shapira: Recommender systems handbook, 2015 - Springer
- S. Huang: Introduction to Recommender System, Online: https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75
- P. Kordík: Machine Learning for Recommender systems, Online: https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Šůstek Martin, Ing.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 26. listopadu 2019

Abstrakt

Tato práce se zabývá především přístupy k sestrojení systému pro doporučování filmů. Je zde obecně popsán princip neuronových sítí a rovněž jsou zde shrnuty základní i pokročilé techniky pro tvorbu doporučovacích systémů. Jádrem práce je návrh, implementace a experimentování se systémem, jehož cílem je doporučování filmů na základě dat pocházejících z volně dostupných datových sad. Pro předpovědi hodnocení, které by uživatel udělil filmům po jejich shlédnutí, systém využívá faktorizační model založený na kolaborativním filtrování. Práce dále řeší souvislosti konfigurace hyperparametrů modelu s přesností doporučení, provádění experimentů za účelem zlepšení přesnosti modelu a nakonec srovnání modelu s existujícími řešeními.

Abstract

This thesis primarily addresses various methods of constructing a system for movie recommendations. Both basic and advanced techniques required for creating a recommender system are also covered in the thesis. The core of the thesis is designing, implementing and experimenting with a system for movie recommendations based upon the data originating from publicly accessible datasets. In order to predict ratings that the user would give to movies after watching them, the system utilizes a factorization model based on collaborative filtering. This thesis also describes the relation between model hyperparameter configuration and prediction accuracy, experiments that were conducted in order to further improve the model accuracy and finally compares the implemented model with existing solutions.

Klíčová slova

doporučovací systémy, neuronové sítě, latentní faktorové modely, filmy, kolaborativní filtrování, doporučování filmů, faktorizace matic, spotlight, datové sady

Keywords

recommender systems, neural networks, latent factor models, movies, collaborative filtering, movie recommendations, matrix factorization, spotlight, datasets

Citace

JANKO, Pavel. *Systém pro doporučování filmů*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Šůstek

Systém pro doporučování filmů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Šůstka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Janko
30. července 2020

Poděkování

Mnohokrát děkuji panu Ing. Martinu Šůstkovi za ochotu a cenné rady, které mi poskytl při zpracovávání této bakalářské práce.

Obsah

Kapitola 1

Úvod

Lidé často spoléhají na doporučení ostatních při každodenním rozhodování o rutinních záležitostech, jako kterou knihu si přečíst nebo na který film se podívat. Zaměstnavatelé zohledňují například doporučení z předchozích firem při hledání nových zaměstnanců. V životě člověk zkrátka čelí nespočet rozhodnutím, a proto vznikla potřeba technologie, která dokáže toto rozhodování uživateli ulehčit a vybrat z nepřeberného množství informací, produktů a služeb pouze ty, které jsou pro uživatele relevantní.

Došlo tedy ke zrodu doporučovacích systémů. Tyto systémy omezují sadu možností, které jsou uživateli nabízeny, pouze na ty, které se vztahují k jeho preferencím nebo předchozímu chování. Cílem této práce je tvorba právě takového systému, který na základě uživatelovy historie hodnocení doporučí pokud možno vyhovující film.

Kapitola 2 se zabývá vysvětlením pojmu doporučovací systémy a jejich významem. Rovněž jsou rozebrány jednotlivé techniky doporučování spolu s jejich klady i zápory, které do velké míry ovlivňují oblast jejich využití. Také je zde popsán princip fungování dvou velice známých komerčních doporučovacích systémů.

Obecným popisem umělých neuronových sítí, jejich trénování a způsobu jejich využití u doporučovacích systémů, se zabývá kapitola 3.

V kapitole 4 se lze dočíst o návrhu doporučovacího systému, se kterým souvisí volba vhodné techniky, kterou lze využít k předpovědi hodnocení pro uživatelem nezhlédnuté filmy. Dále se v ní vyskytuje srovnání vybraných open-source knihoven určených pro strojové učení a popisem, jakým způsobem jsou v aplikaci využity. Nakonec je zde rozebrán proces ladění hyperparametrů modelu, který vedl k jejich konečné konfiguraci.

V rámci 5. kapitoly popisuji provedené experimenty s faktorizačním modelem, jejichž cílem bylo další zlepšení přesnosti modelem předpovězených hodnocení.

O dosažených výsledcích implementovaného řešení a srovnání s ostatními řešeními nad stejnou datovou sadou se lze dočíst v kapitole 6.

V závěru se nachází shrnutí výsledků a z nich vyplývajícího přínosu práce, nových poznatků a výčet případných vylepšení, které by mohly systém dále obohatit.

Kapitola 2

Doporučovací systémy

Doporučovací systémy jsou softwarové nástroje a techniky, které poskytují návrhy položek, které by mohly s největší pravděpodobností uživatele zajímat. Tyto návrhy souvisí s činnostmi, které vyžadují, aby se uživatel určitým způsobem rozhodl. Například jakou hudbu si poslechnout, jaký článek si přečíst nebo který dárek zakoupit. Typicky se doporučovací systém zaměřuje pouze na jednu z těchto kategorií a odpovídajícím způsobem se tomu přizpůsobí návrh, uživatelské rozhraní i technika doporučování, kterou systém implementuje. [24]

Položky v rámci doporučovacích systémů lze charakterizovat například jejich složitostí, hodnotou, nebo užitečností. Hodnota položky může být pozitivní, pokud je pro uživatele relevantní, naopak může být negativní, když se projeví jako nevhodná pro uživatele. Příkladem položek s nízkou úrovní složitosti mohou být například novinky, webové stránky, knihy, či filmy. Mezi složitější položky pak lze zařadit elektroniku, finanční investice a pracovní nabídky.

Položky (v této práci představují filmy) a uživatele (obecně jakékoliv entity) spojují interakce, které v sobě obsahují důležité informace o vztahu mezi uživatelem a položkou. Asi nejpopulárnější způsob shromažďování dat o interakcích je formou hodnocení [10]. Tento proces je buď explicitní, kdy je uživatel přímo dotázán na jeho názor na určitou položku, nebo implicitní, kdy se tyto informace odvozují od uživatelovy činnosti. V dnešních pokročilých systémech se můžeme také setkat s kombinací těchto přístupů.

Podle [20] mezi hlavní důvody využití doporučovacích systému řadíme:

- Zvýšení prodeje díky doporučování vhodných položek, které zpravidla reflektují uživatelovy potřeby.
- Prodej méně známých položek, na které by uživatel bez doporučení nemusel nikdy narazit.
- Zlepšení zážitku uživatele tím, že mu systém poskytne relevantní a zajímavá doporučení, které často souvisí s přívětivějším uživatelským rozhraním.
- Udržování uživatelů, jelikož uživatelé mají tendence být loajální vůči aplikaci, která si jej pamatuje a přizpůsobuje se mu.

V kontextu doporučovacích systémů se objevuje pojem doporučovací techniky. Mezi nejčastěji využívané techniky se řadí **kolaborativní filtrování** (*collaborative filtering*) a **doporučování založené na obsahu** (*content-based filtering*), přičemž současně využívané

systémy často kombinují oba tyto způsoby doporučování [4]. Další technika, která se namísto hodnocení zaměřuje na uživatelovu historii aktivity, je známá jako **doporučování založené na sezení** (*session-based recommendations).*

2.1 Kolaborativní filtrování

Kolaborativní filtrování je technika, pomocí které je možné doporučovat položky uživatelům na základě hodnocení položek ostatními uživateli. Funguje pomocí vyhledání oblíbených položek malé skupiny uživatelů, kteří mají podobný vkus jako uživatel, kterému chceme film doporučit. Tyto položky jsou poté seřazeny podle odhadu hodnocení.

Pro kolaborativní filtrování existují dva hlavní přístupy. Prvním je přístup založený na paměti, který využívá statistických metod, zatímco druhý přístup je založen na modelech pro strojové učení.

2.1.1 Přístup založený na paměti

Tento přístup lze rozdělit na dva hlavní druhy. Prvním je filtrování mezi uživateli, které funguje na základě vyhledání uživatelů s podobnými hodnoceními. Druhý způsob funguje na základě doporučování položek, které se podobají těm, které uživatel v minulosti ohodnotil.

Kolaborativní filtrování mezi uživateli

U kolaborativního filtrování mezi uživateli jsou na celou datovou sadu aplikovány statistické techniky pro získání předpovědi hodnocení $e_{i,j}$ udělené položce j uživatelem i. Proces předpovědi se skládá ze dvou kroků:

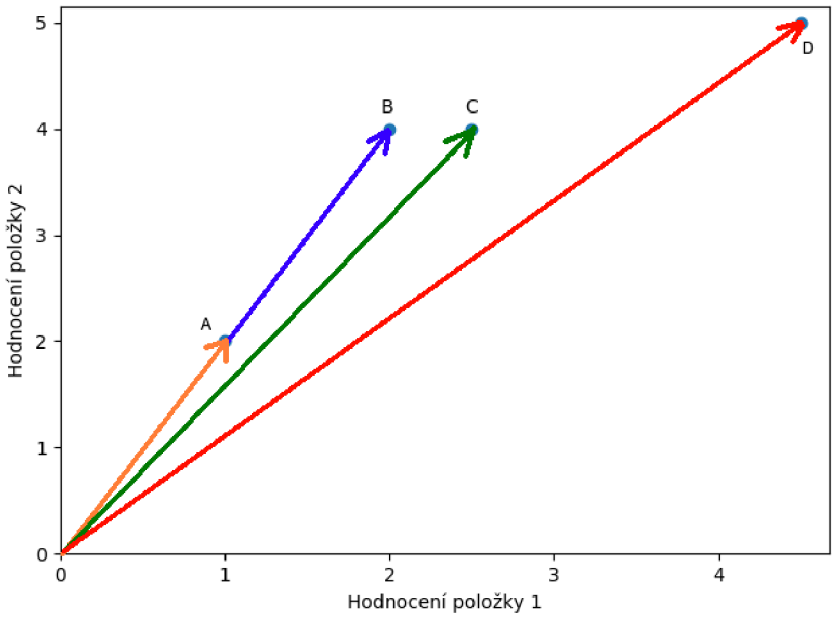
1. Vyhledání uživatelů, kteří hodnotí podobně jako uživatel i a kteří hodnotili položku j
2. Výpočet předpovědi hodnocení $e_{i,j}$ na základě uživatelů získaných v předchozím kroku

Na Obrázku 2.1, kde uživatelé představují jednotlivé body a vektory v 2D prostoru, je ilustrován princip vyhledání podobných uživatelů. Za podobnost uživatelů mezi sebou lze v tomto případě považovat euklidovskou vzdálenost bodů reprezentující uživatele na obrázku. Další možnost, jak podobnost určit, je využít **kosinovou podobnost** (*cosine similarity*), jejíž výstup je v intervalu od -1 do 1 se zmenšujícím se rozdílem mezi úhly, které svírají vektory uživatelů. Pro pouze kladný vstup, jako tomu je na Obrázku 2.1, je výstup kosinové podobnosti v intervalu od 0 do 1. Nechť R je matice hodnocení, kde se neznámá hodnocení (kdy uživatel film nehodnotil) rovnají 0. Vztah pro výpočet podobnosti touto funkcí mezi vektory hodnocení uživatelů R_i a R_j, kde m je počet položek a $R_{i,k}$ a $R_{j,k}$ představují hodnocení položky k těmito uživateli, je následující:

$$similarity(R_i, R_j) = \frac{\sum_{k=1}^{m} R_{i,k} R_{j,k}}{\sqrt{\sum_{k=1}^{m} R_{i,k}^2 \sum_{k=1}^{m} R_{j,k}^2}}, \tag{2.1}$$

kde při dělení 0 není kosinová podobnost definována. Z Obrázku 2.1 lze dále vypozorovat, že v případě, že bychom využili pro určení podobnosti euklidovskou vzdálenost, tak si budou nejvíce podobní uživatelé B a C. Avšak pokud se rozhodneme využít úhel mezi vektory, tak vzhledem k tomu, že vektory uživatelů A a B svírají stejný úhel, tak jsou si podle této

metriky více podobní než uživatelé B a C, přestože mají rozdílná hodnocení. V kontextu doporučování filmů může důvodem být například to, že uživatel A je filmový kritik, který uděluje vždy nižší hodnocení než průměrný uživatel, navzdory tomu, že se film oběma uživatelům líbil stejně.



Obrázek 2.1: Hodnocení dvou položek uživateli A, B, C a D.

Za účelem zohlednění individuálních preferencí jednotlivých uživatelů je možné před určením podobnosti odečíst průměrné hodnocení uživatele od všech hodnocení, které položkám udělil. Tento princip je obecně znám pod pojmem **Pearsonova korelace** (*Pearson correlation*). Pokud využijeme stejná označení veličin, jako ve vzorci 2.1, přičemž $\overline{R_i}$ a $\overline{R_j}$ značí průměrná hodnocení uživatelů i a j, tak lze **Pearsonův korelační koeficient** dvou vybraných uživatelů vyjádřit následovně:

$$\rho(R_i, R_j) = \frac{\sum_{k=1}^{m}(R_{i,k} - \overline{R_i})(R_{j,k} - \overline{R_j})}{\sqrt{\sum_{k=1}^{m}(R_{i,k} - \overline{R_i})^2 \sum_{k=1}^{m}(R_{j,k} - \overline{R_j})^2}}, \tag{2.2}$$

kde při dělení 0 koeficient opět není definován. Po získání podobných uživatelů uživateli je nutné určit odhad hodnocení $e_{i,j}$, kde i je uživatel a j je položka. Stejně jako podobnost je možné hodnocení určit vícero způsoby. Nejjednodušší způsob je vypočítat průměrné hodnocení prvních n nejpodobnějších uživatelů uživateli i. Nechť U_i je množina uživatelů a $U_{i,k}$ je k-tý nejpodobnější uživatel uživateli i. Výpočet pak vypadá následovně:

$$e_{i,j} = \frac{1}{n}\sum_{k=1}^{n} R_{u,j}, \quad \text{kde } u \ = U_{i,k} \tag{2.3}$$

Odhad hodnocení $e_{i,j}$ nás zajímá právě tehdy, pokud nám pro danou kombinaci uživatele a filmů v původní matici hodnocení R chybí hodnocení. Nastanou však situace, kdy jsou například první 2 uživatelé podobnější uživateli i než následující uživatelé. V takovém případě je vhodné k hodnocení uživatelů přidat váhový koeficient, který nám umožní

tuto informaci při výpočtu zahrnout. Čím větší bude podobnost uživatele k uživateli i, tím větší bude hodnota koeficientu. Pro určení podobnosti zde lze využít například kosinovou podobnost (viz 2.1). Upravený vztah lze zapsat takto:

$$e_{i,j} = \frac{1}{n} \frac{\sum_{k=1}^{n} R_{u,j} similarity(R_i, R_k)}{\sum_{k=1}^{n} similarity(R_i, R_k)}, \text{ kde } u = U_{i,k} \tag{2.4}$$

Při využití kolaborativního filtrování mezi uživateli je možné se setkat s problémem, kdy pokud oba uživatelé hodnotili pouze jeden společný film stejným hodnocením, tak by jejich vypočítaná podobnost v některých případech mohla být vyšší oproti podobnosti s uživateli, kteří jsou jim opravdu podobní. Tento problém se zpravidla řeší tak, že se hledají pouze podobní uživatelé s více společnými hodnoceními, přičemž konkrétní počet společných hodnocení se odvíjí od celkového množství filmů. [15]

Kolaborativní filtrování mezi položkami

U kolaborativního filtrování mezi uživateli se pokoušíme vyhledat n uživatelů podobných uživateli i a předpovědět jeho hodnocení $e_{i,j}$ položky j na základě průměru hodnocení položky ostatními uživateli. U kolaborativního filtrování mezi položkami je však cílem vyhledat n položek podobných položce p, pro kterou chceme provést předpověď hodnocení $e_{p,q}$ uživatelem q, přičemž výsledný odhad hodnocení dané položky je roven průměru hodnocení podobných položek uživatelem. Nechť I_p je množina položek a $I_{p,k}$ je k-tá nejpodobnější položka položce p. Odhad hodnocení zahrnující kosinovou podobnost můžeme zapsat následovně:

$$e_{p,q} = \frac{1}{n} \frac{\sum_{k=1}^{n} R_{i,q} similarity(R_p, R_k)}{\sum_{k=1}^{n} similarity(R_p, R_k)}, \text{ kde } i = I_{p,k} \tag{2.5}$$

V systému, kde je větší počet uživatelů než položek, je filtrování mezi položkami zpravidla rychlejší a stabilnější než filtrování mezi uživateli. Je to dáno především nižší proměnlivostí průměrného hodnocení položky v porovnání s proměnlivostí průměrného hodnocení položek uživatele.

Navzdory tomu ale filtrování mezi položkami dosahuje horších výsledků než filtrování mezi uživateli pro datové sady například z oblasti filmů, či seriálů. Důvodem je především předvídatelnost doporučení – většinou jsou doporučovány filmy stejného žánru s podobným hereckým obsazením. U těchto datových sad jsou zaznamenány lepší výsledky při využití přístupu založeného na modelech, či hybridních doporučovacích systémů. [11]

2.1.2 Přístup založený na modelech

Při tomto přístupu jsou modely pro kolaborativní filtrování založené na algoritmech pro **strojové učení**, které jsou rozebrány v kapitole 3. Výstupem těchto algoritmů je předpověď uživatelova hodnocení pro jednotlivé položky. Doporučovací systémy založené na modelech mají oproti paměťovému přístupu několik výhod.

První výhodou je lepší využití paměti, jelikož je při provádění doporučení u tohoto přístupu zapotřebí v paměti udržovat pouze výsledný model, který je zpravidla mnohonásobně menší než původní matice hodnocení R. Další rozdíl nalezneme ve fázi předzpracování dat, kdy při využití paměťových přístupů je časová náročnost kvadratická, u filtrování mezi uživateli podle počtu uživatelů a filtrování mezi položkami zase podle počtu položek. V případě systémů založených na modelech je předzpracování izolováno od doporučování, protože lze

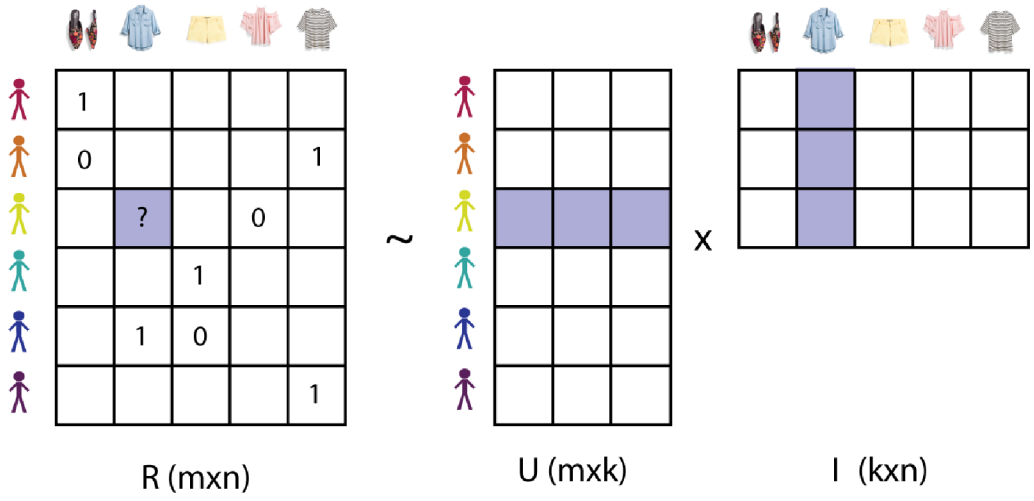
pro doporučování použít již předem natrénovaný serializovaný model, takže jsme schopni výsledky získat rychleji.

Algoritmy využívající tento přístup lze podle [10] rozdělit do několika skupin, v práci se budu zaměřovat na skupinu využívající **faktorizaci matic**.

Faktorizace matic

Faktorizaci matic lze chápat jako rozložení větší matice na součin dvou menších matic. Mějme matici R složenou z m řádků a n sloupců. Tuto matici je možné rozložit na součin dvou matic U a I o velikost $m \times k$ a $k \times n$, kde počet sloupců matice U je roven počtu řádků matice I.

U doporučovacích systémů lze tímto způsobem rozložit matici hodnocení R na matici uživatelů a matici položek, kde m řádků matice U představuje m vektorů uživatelů popsatelných k charakteristikami (*embeddings*), a n sloupců matice I zase n vektorů položek popsatelných k charakteristikami, což je znázorněno na Obrázku 2.2. Neznámé hodnoty v matici (tedy chybějící hodnocení, kdy uživatel nehodnotil položku) můžeme v programovacích jazycích například nahradit konstantou `NaN`. Formálně však faktorizace matic znamená, že $U \cdot I = R$, ale jelikož jsou v matici R chybějící hodnoty, tak platí $U \cdot D = E$, kde E je matice odhadů hodnocení. Cílem je poté vyhledat takové matice U a I, aby pro každý element matice R, který má definovanou hodnotu, byla celková chyba přes všechny uživatele a položky oproti matici odhadů hodnocení E minimální.

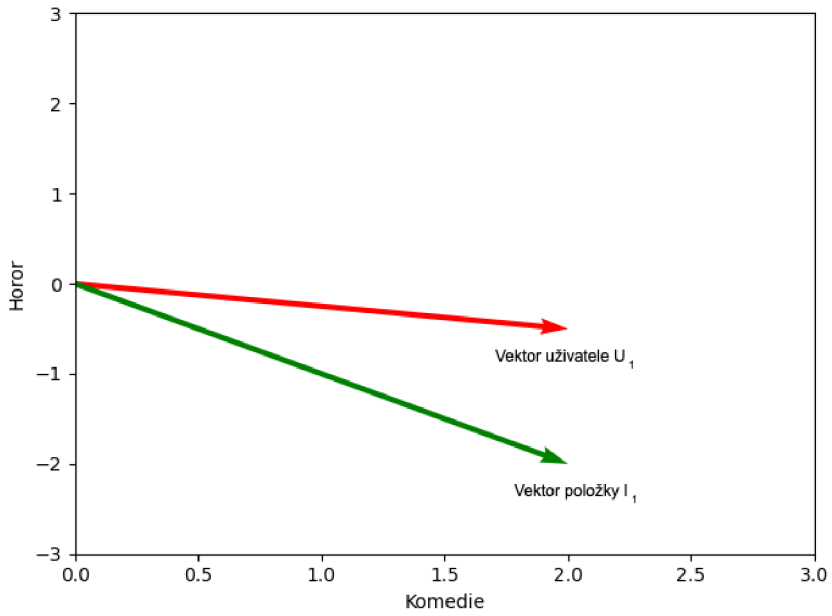


Obrázek 2.2: Princip faktorizace matic v kontextu doporučovacích systémů. Převzato z [2] a upraveno.

Význam jednotlivých charakteristik uživatelů a položek je před námi skryt. Pokud by byla dimenzionalita $k = 2$ (počet složek vektorů uživatelů a položek), tak je možné pro popis principu využití faktorizace matic u doporučování filmů na interpretaci charakteristik jednoduše nahlížet následovně:

- Vektor uživatele $U_i = (z_1, z_2)$ vyjadřuje, do jaké míry má uživatel i rád komedie a horory

- Například vektor uživatele $U_1 = (2, -0.5)$ by tedy znamenal, že uživatel má v oblibě komedie, ale horory naopak spíše neupřednostňuje
- Vektor filmu $I_j = (z_1, z_2)$ zase vyjadřuje do jaké míry je film j komedie a do jaké míry zase horor
- Vektor filmu by tedy mohl nabývat hodnot $I_1 = (2, -2)$, což by znamenalo, že se jedná o komedii bez hororových prvků
- Po vynásobení vektoru uživatele s vektorem filmu získáme odhad hodnocení uživatele tohoto filmu, tedy $(2, -0.5) \cdot (2, -2) = 5$



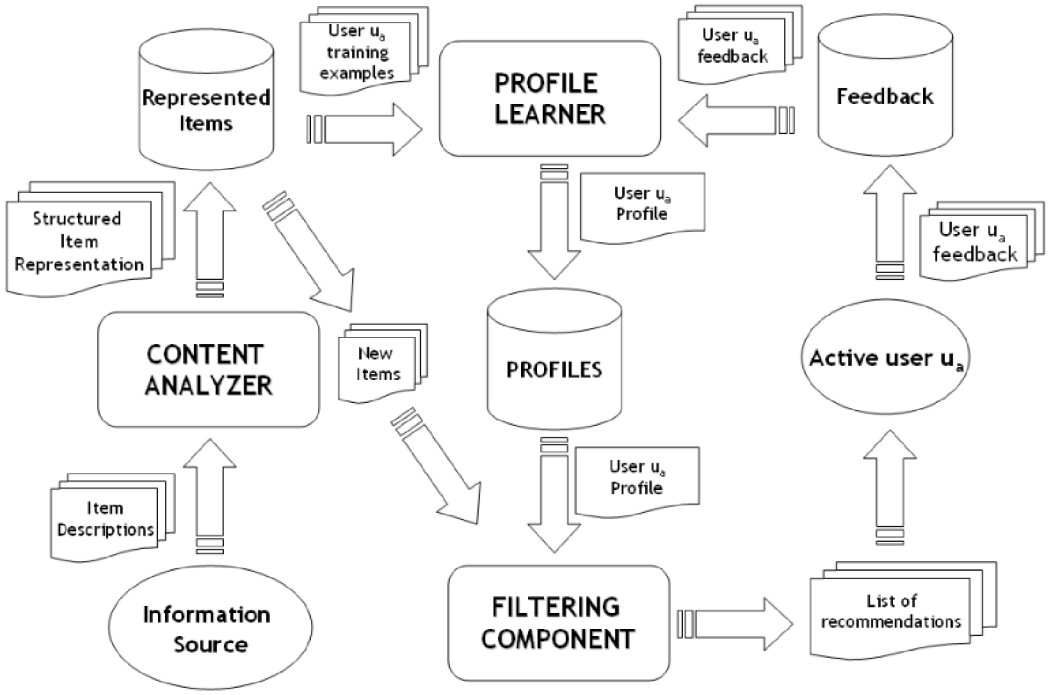
Obrázek 2.3: Reprezentace vektorů uživatele a filmu, kde vektor uživatele U_1 určuje jak moc má rád filmový žánr a vektor filmu I_1 zase do jaké míry se ve filmu vyskytují prvky tohoto žánru. Dimenzionalita $k = 2$.

Reprezentace vektorů uživatele a filmu z příkladu je zachycena na Obrázku 2.3. Ve skutečnosti je však interpretace těchto charakteristik zpravidla daleko složitější a obsáhlejší. Také v příkladu pro vysvětlení pracujeme s pouhými dvěma charakteristikami, obvykle však počet charakteristik dosahuje řádu stovek. Obecně lze říci, že s narůstajícím počtem charakteristik jsou poskytovaná doporučení více personalizovaná. Je ovšem nutné vyhnout se přetrénování modelu, o kterém je možné zjistit více v sekci 4.2.1.

2.2 Doporučování založené na obsahu

Systémy implementující doporučování založené na obsahu analyzují položky, které v minulosti uživatel hodnotil, a sestaví uživatelský profil na základě vlastností těchto položek. Proces předpovědi hodnocení pak spočívá v nalezení položek s podobnými vlastnostmi těm vlastnostem, které jsou zahrnuty v uživatelově profilu. Tyto systémy samozřejmě vyžadují

využití správných technik pro odvození vlastností položek a sestrojení uživatelova profilu. Architektura takového systému je ukázána na Obrázku 2.4.



Obrázek 2.4: Architektura doporučovacího systému založeném na obsahu. Převzato z [19].

Proces doporučení se skládá ze tří hlavních kroků, které vykonávají oddělené komponenty [19]. V prvním kroku analyzátor obsahu převede nestrukturované informace do požadované podoby, se kterou dále pracují zbylé komponenty. V dalším kroku jsou shromážděny data o uživatelových preferencích a na základě nich je vytvořen profil uživatele. Tento profil může například tvořit seznam položek, na které uživatel v minulosti reagoval kladně a nebo záporně ve formátu, který poskytuje analyzátor obsahu. Nakonec filtrovací modul využije uživatelův profil pro porovnání s jednotlivými položkami a jeho výstupem je sestupně seřazený seznam položek podle míry relevance pro daného uživatele. Pro porovnání podobnosti uživatele a položek jsou zpravidla využívány metriky jako kosinová podobnost, viz 2.1.

Doporučovací systémy tohoto typu mají oproti kolaborativnímu filtrování některé výhody, především v případě doporučování nových položek. Pokud nemá položka k dispozici žádná hodnocení, tak jej systém využívající kolaborativní filtrování nikdy nemůže doporučit. Avšak vzhledem k tomu, že metody doporučování založeného na obsahu využívají vlastnosti položek, stačí pouze najít položky s podobnými vlastnostmi, které ostatní uživatelé již v minulosti hodnotili, a doporučovat podle těchto dat.

V porovnání s kolaborativním filtrování přinášejí ale také některé nevýhody. Přestože jsou tyto systémy lepší při doporučování nových položek, dosahují horších výsledků při doporučování položek pro nové uživatele, jelikož pro něj vyžaduje historii hodnocení [11]. Tento problém lze do určité míry řešit například požádáním nového uživatele o explicitní informace o jeho preferencích. Dalším problémem je, že někdy uživateli není položka doporučena pouze proto toho, že není podobná jiným položkám, které v minulosti hodnotil, i když by se uživateli mohla daná položka líbit. Systémy využívající kolaborativní filtro-

vání totiž tomuto problému předcházejí zohledňováním preferencí uživatelů s podobným vkusem. Asi hlavní nevýhodou tohoto přístupu je však náročnost implementace, která je v porovnání s kolaborativním filtrováním výrazně složitější [19].

2.3 Doporučování založené na sezení

Na rozdíl od předchozích přístupů, které se buď snaží předpovědět co nejpřesněji uživatelova hodnocení pro položky a nebo nalézt položky podobné těm, které uživatel už hodnotil, se u doporučování založeného na sezení pokoušíme určit na základě **posloupnosti interakce s položkami** jinými uživateli právě takovou položku, se kterou by uživatel interagoval s největší pravděpodobností jako další. Předpokládáme tedy, že pokud například velké množství uživatelů interagovalo v postupném pořadí s položkou A a poté s položkou B, tak nový uživatel po interakci s položkou A bude také chtít s největší pravděpodobností interagovat s položkou B.

Zřejmě největší výhodou při využití tohoto přístupu je schopnost provádět relativně přesná doporučení i bez většího množství dat shromážděných o novém uživateli. Existující doporučovací systémy totiž často musejí doporučení provádět na základě dat získaných během krátké doby (například webová stránka pro rybářské náčiní), oproti systémům, které mají k dispozici daleko více informací o uživatelových historických interakcích. Tento problém se zpravidla řeší implementací doporučování založeném na obsahu (viz 2.2), kdy jsou doporučovány podobné položky. Pokud by ale bylo namísto toho využito doporučování založené na sezení, kdy je zohledňováno pořadí akcí prováděných uživatelem během jeho krátké přítomnosti na stránce, je možné dosáhnout vyšší přesnosti při doporučování [14].

2.4 Komerční systémy

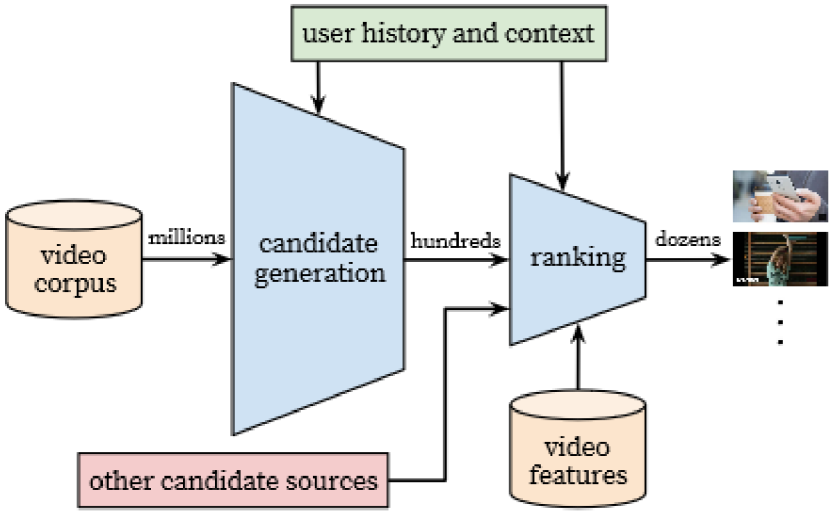
V této sekci se lze dočíst o architektuře doporučovacích systémů implementovaných v rámci komerčních řešení, konkrétně firmami *YouTube*[1] a *Netflix*[2], které část informací o principu fungování svých systémů zveřejnily. Tyto firmy se na trhu pohybují už několik let a shromažďují jak explicitní, tak implicitní data o uživatelích. Tudíž mají k dispozici datové sady obrovských rozměrů, které jsou mnohonásobně větší a obsahují více metadat, než volně dostupné datové sady pro účely výzkumu.

Mimo jiné *YouTube* ani *Netflix* neposkytuje nezávislý systém, který by filmy či videa doporučoval samostatně. Jedná se pouze o část (i když nedílnou) jejich online streamovací platformy. Získávání implicitních dat je tedy pro tyto systémy jednoduché, protože uživatelé interagují s doporučovaným obsahem přímo v rámci aplikace. Je tedy možné shromažďovat informace o tom, které položky uživatel vyhledával, které položky má v poslední době nejraději a podobně. Tento prakticky nespočetný objem dat společnostem umožňuje vynahradit určité technologické nedostatky jejich doporučovacích systémů.

2.4.1 YouTube

Doporučovací systém na stránce *YouTube* sestává ze dvou neuronových sítí, kde jedna generuje kandidáty pro doporučení a druhá tyto kandidáty ohodnocuje (viz Obrázek 2.5). [8]

[1] https://www.youtube.com/
[2] https://www.netflix.com/



Obrázek 2.5: Architektura doporučovacího systému využívaného na YouTube. Převzato z [7].

Generování kandidátů

Síť, která generuje kandidáty, je postavena na principu kolaborativního filtrování popsaného v kapitole 2.1. Čerpá data z uživatelovy historie aktivity na stránce a využívá je jako vstup. Výstupem jsou pak stovky videí, které by mohly uživatele zajímat.

Asi nejdůležitějším prvkem při doporučování *YouTube* videí je zaměření se na nově nahraný obsah. Výzkumy opakovaně ukazují, že uživatelé upřednostňují novější obsah, avšak nikoliv za cenu relevance vůči jejich zájmům. Kromě doporučování novinek je rovněž důležité do doporučování zahrnout virální (*viral*) videa, která jsou v konkrétním okamžiku v uživatelově demografické skupině vysoce populární. [7]

Data o historii zhlédnutí poskytována doporučovacímu systému jsou vybírána ze zhlédnutí všech videí, nejen těch často doporučovaných. V opačném případě by se nový obsah objevoval jen s těží a doporučovací systém by byl velmi předpojatý vůči některým videím [8]. Pokud uživatelé naleznou videa i jiným způsobem než prostřednictvím doporučení, je nutno tento objev rychle propagovat ostatním uživatelům prostřednictvím kolaborativního filtrování.

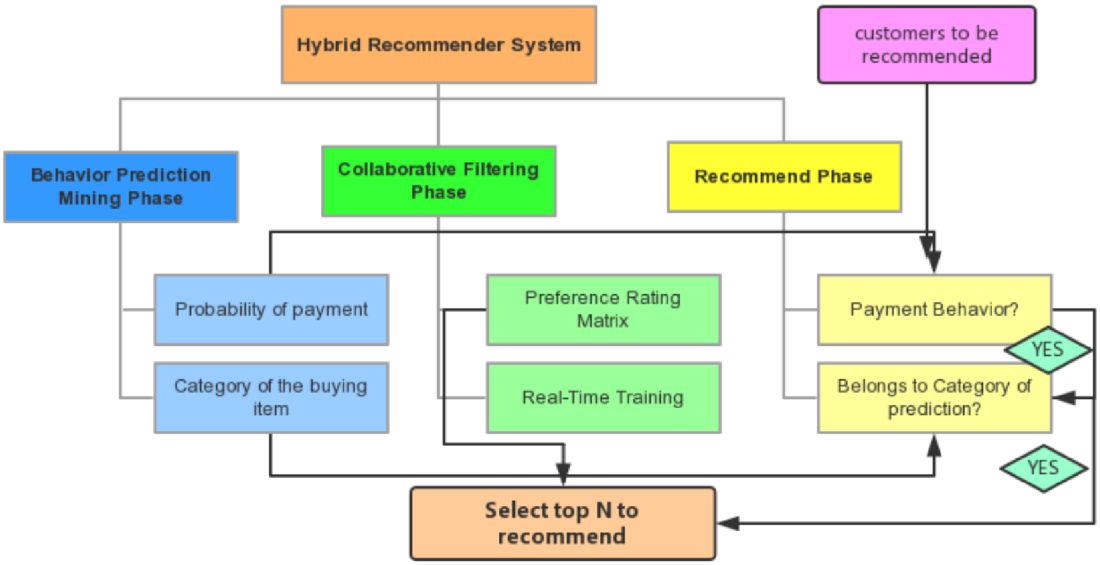
Hodnocení kandidátů

Za účelem poskytnutí co nejlepších doporučení je však zapotřebí využít druhou neuronovou síť, která videa, jež jsou výstupem z první neuronové sítě, seřadí dle číselného hodnocení, které se odvíjí zejména od detailů o uživateli a jednotlivých videích [7]. Tento přístup zaručuje, že jsou doporučení vybírána z opravdu širokého rozsahu a zároveň jsou vybírána jen ta, která by mohla být pro uživatele zajímavá.

V modelu pro hodnocení kandidátů jsou využívány stovky vlastností, přičemž největší obtíží je určit, jak posloupnost uživatelských akcí souvisí s hodnocením videa. Z pozorování vyplývá, že nejdůležitější signály jsou například počet videí, které uživatel zhlédnul na konkrétním *YouTube* kanále, nebo kdy naposled uživatel zhlédnul video na toto téma.

2.4.2 Netflix

Průzkum spotřebitelů ukazuje, že průměrný předplatitel *Netflixu* ztrácí zájem o sledování po 60 až 90 sekundách vybírání, přičemž prozkoumal mezi 10 a 20 tituly. Proto je naprosto klíčové, aby v této době uživatel našel takový titul, který jej zajímá, a pro splnění tohoto účelu využívá *Netflix* široký rozsah algoritmů tvořící komplexní doporučovací systém. [12]



Obrázek 2.6: Architektura hybridního modelu využívaného firmou *Netflix*. Převzato z [11].

Doporučovací systém je využit převážně na hlavní stránce, která se skládá ze sekcí (řádků), jako jsou žánry, nejlepší výběry a v neposlední řadě například populární tituly. Architektura tohoto systému je zachycena na Obrázku 2.6, přičemž detailním popisem jeho částí se v práci zabývat nebudu. Uživatelé, kteří se rozhodnou zhlédnout určitý titul, pochází z 80% právě z hlavní stránky a zbylých 20% pak filmy a seriály vybírá na základě vyhledávání názvů a klíčových slov [1].

Personalizovaný výběr

Algoritmus pro personalizovaný výběr se využívá právě v sekcích, které souvisí s konkrétními žánry. Funguje na základě seřazení celého katalogu videí podle konkrétního žánru pro každého předplatitele. Toto seřazení je převážně unikátní a zohledňuje zejména uživatelovu historii zhlédnutí, konkrétně zajímá-li se uživatel více o filmy než seriály. Dále bere v úvahu oblíbené herce či režiséry a podobně. Schopnost algoritmu poskytovat osobní doporučení je však z části limitována popularitou určitých titulů. [1]

Trendy

Na stránce rovněž existuje sekce zaměřující se na dočasné trendy v rozsahu několika minut až po dny. Tyto trendy jsou velmi efektivní pro předpověď potencionálně zajímavých titulů pro uživatele, zejména v kombinaci s malou mírou personalizace [12]. Doporučovací algoritmus identifikuje zejména takové trendy, které se opakují každých pár měsíců, ale mají pouze krátkodobý efekt (například zvýšený zájem o romantické filmy poblíž dne svatého

Valentýna). Dále bere v úvahu jednorázové nedlouho trvající události, které jsou populární zejména v médiích, či na internetu (ku příkladu blížící se hurikán zvyšuje zájem o dokumenty pojednávající o této tématice, či jiných přírodních katastrofách).

A/B testování

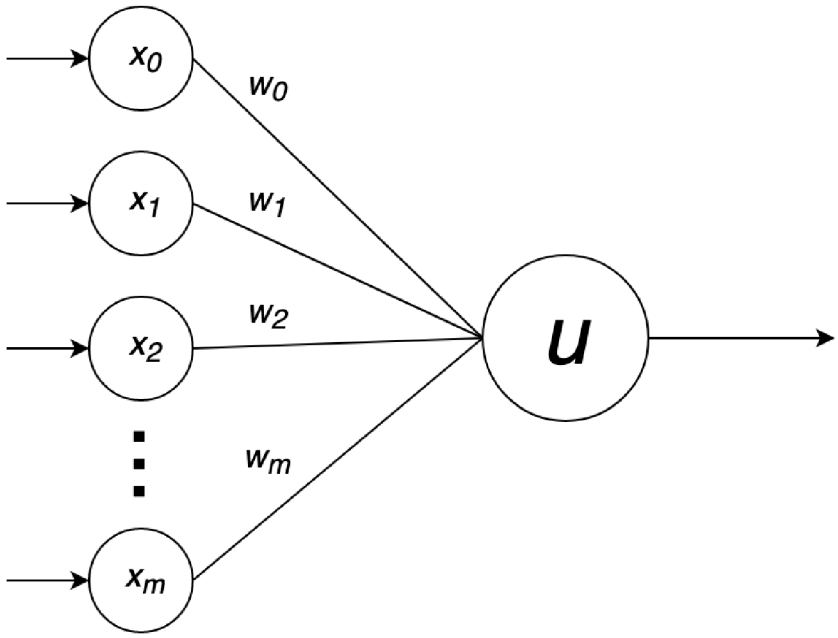
Jelikož má *Netflix* velký počet předplatitelů, je jednoduché testovat různé konfigurace modelu pomocí A/B testování. Obecně se jedná o experiment, při kterém jsou testovací subjekty rozděleny do dvou skupin a každé skupině je prezentován prakticky stejný obsah, až na jednu proměnnou. Po uběhnutí předem stanoveného časového úseku jsou výsledky obou variant srovnány a na základě této analýzy se poté určí, která varianta je vhodnější.

Způsob realizace tohoto typu testování na streamovací platformě *Netflix* pak probíhá tak, že jsou předplatitelé rozděleni do několika buněk, kde doporučování uživatelům v rámci každé buňky zprostředkovává jen minimálně upravený doporučovací model. Poté jsou sledovány metriky jako délka pobytu na stránce, délka výběru pořadu či filmu ke sledování a podobně. Na základě těchto metrik je poté vybrán takový model, který poskytoval při doporučování obsahu nejrelevantnější doporučení. Zajímavý je tedy fakt, že testování je prováděno za běhu aplikace na skutečných uživatelích, nikoliv vybraných testovacích subjektech, a vede k drastickým vylepšením přesnosti doporučení, které se odráží na neustálém růstu platformy. [12]

Kapitola 3

Umělé neuronové sítě

Umělé neuronové sítě jsou inspirovány biologickými neurony, které se jsou základním stavebním prvkem nervové soustavy. Hlavní částí těchto sítí jsou propojené uzly zvané neurony. Na umělý neuron můžeme nahlížet jako na matematickou funkci s jediným výstupem y a m vstupy x, přičemž hodnota každého z těchto vstupů je modifikována váhovým koeficientem w, jehož hodnota se upravuje během fáze učení sítě. Každý neuron může rovněž obsahovat uzel pro předpětí (*bias*) x_0. Na Obrázku 3.1 vidíme obecnou podobu takového neuronu.



Obrázek 3.1: Model umělého neuronu.

Matematicky lze tedy výstup jednoho neuronu zapsat následovně:

$$y = g(u) = g(f(\vec{x}, \vec{w})), \tag{3.1}$$

kde f představuje bázovou funkci a g funkci aktivační [27]. Nejčastěji využívaná bázová funkce je podle [13] lineární bázová funkce (LBF), zapsáno:

$$u = \sum_{i=1}^{m} x_i w_i = \vec{x} \cdot \vec{w}, \tag{3.2}$$

Místo skalárního součinu použitého v rovnici 3.2 je také možné použít součin maticový, kdy jsou oba vektory $\vec{x}$ a $\vec{w}$ považovány za sloupcové (transponovaný vektor $\vec{x}^T$ je tedy řádkový):

$$u = \vec{x}^T \vec{w}, \tag{3.3}$$

Neurony jsou zpravidla organizovány do více vrstev. Mezi vstupem a výstupní vrstvou se nachází libovolný počet skrytých vrstev (může být i nulový). V případě, že jsou neurony jedné vrstvy propojeny vždy pouze s neurony přímo následující vrstvy, tak tvoří řízený acyklický graf a jsou známy jako dopředné neuronové sítě [28]. Rovněž existují rekurentní neuronové sítě, které umožňují propojení neuronů ve stejné vrstvě a nebo propojení s neurony kterékoliv předchozí vrstvy [21].

3.1 Trénování umělých neuronových sítí

Při trénování neuronová síť pro každý vstup vypočítá výstup. Ztrátová funkce tento výstup vyhodnotí a optimalizátor poté na základě ztrátové funkce upraví parametry sítě tak, aby se hodnota ztrátové funkce zmenšila. Před začátkem trénování jsou parametry sítě zpravidla inicializovány náhodně podle nějakého rozložení. Existuje více způsobů, jak učit neuronovou síť, přičemž v práci se budu zabývat pouze učením s učitelem, protože tento způsob učení je využit implementovaným modelem.

Při trénování modelů se dále setkáváme s tzv. **hyperparametry**. Jedná se o hodnoty ovlivňující průběh trénovací fáze modelu, jejichž vhodnou úpravou lze dosáhnout lepších výsledků při evaluaci modelu. Na rozdíl od parametrů sítě, které jsou upravovány v průběhu trénování sítě, jsou hyperparametry nastaveny před zahájením trénování.

Učení s učitelem

Při učení s učitelem má síť k dispozici datovou sadu s ukázkovými daty. Trénovací data sestávají z dvojic vstupů a požadovaných výstupů. Algoritmus při učení s učitelem analyzuje trénovací data a poté poskytne odvozenou funkci, kterou lze použít pro předpověď výstupní hodnoty pro každý platný vstup [22].

Datová sada se zpravidla rozděluje na tři části, přičemž konkrétní poměr rozdělení se podle [9] může lišit v závislosti na velikosti datové sady nebo případu užití:

1. **Trénovací sada** – obvykle 70% dat, slouží pro trénování neuronové sítě
2. **Validační sada** – obvykle 15% dat, slouží pro vyhodnocení modelu nezávisle na trénovacích datech za účelem dalšího ladění hyperparametrů (viz 4.2)
3. **Testovací sada** – obvykle 15% dat, slouží ke konečnému vyhodnocení modelu na dosud neviděných datech

Počet epoch pak určuje počet průchodů datové sady při trénování. Jelikož je při přechodu mezi epochami zachováno nastavení parametrů neuronové sítě, tak je opakováním

trénovacího procesu modelu dalšími úpravami parametrů umožněno potencionálně zvýšit přesnost doporučení. Při volbě příliš velkého počtu epoch však může dojít k tzv. přetrénování, při kterém lze pozorovat vysokou přesnost na trénovací sadě, avšak při evaluaci modelu na testovacích datech odchylka doporučení značně narůstá. Pokud chceme stanovením velkého počtu epoch umožnit modelu rozpoznávat v datech komplexnější vzory (*patterns*), ale dochází poté k přetrénování, tak lze tento problém řešit zavedením regularizace, o které je možné dočíst se více v sekci 4.2.1.

Proces trénování neuronové sítě zpravidla končí poté, co je dosažen předem stanovený počet epoch a nebo například v případě, že nedošlo ke změně hodnoty ztrátové funkce v posledních několika epochách na validační sadě. [28]

3.1.1 Evaluační metriky

Evaluační metriky slouží k vyhodnocování modelu. Mezi nejčastěji využívané metriky řadíme MAE (*mean absolute error*), definovanou jako průměrný absolutní rozdíl mezi předpovězenými a skutečnými hodnoceními, zapsáno:

$$\text{MAE} = \frac{\sum_{u,i} |e_{u,i} - R_{u,i}|}{N}, \tag{3.4}$$

kde $e_{u,i}$ je předpověď hodnocení položky i uživatelem u, $R_{u,i}$ je skutečné hodnocení a N je celkový počet hodnocení v testovací sadě.

Další často užívanou metrikou je MSE (*mean square error*), značící střední kvadratickou odchylku. Tato klade větší důraz na vyšší hodnoty absolutních chyb. Její zápis je následující:

$$\text{MSE} = \frac{\sum_{u,i} (e_{u,i} - R_{u,i})^2}{N} \tag{3.5}$$

Metrika MSE je mimo jiné mezikrokem k výpočtu RMSE (*root mean square error*), kdy pouze odmocníme MSE:

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{3.6}$$

Všechny zmíněné evaluační metriky je rovněž možné použít jako ztrátové funkce, jejichž hodnotu se snažíme minimalizovat v průběhu trénování neuronové sítě.

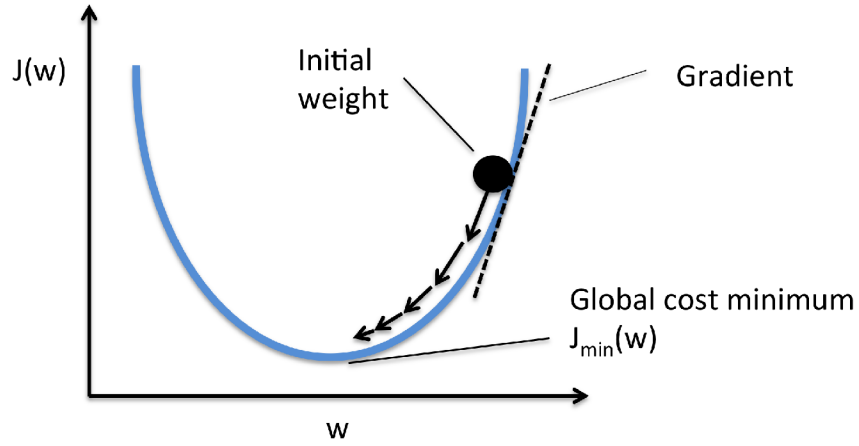
3.1.2 Optimalizátory

V průběhu trénování neuronové sítě se její parametry upravují za účelem nalezení pokud možno minima ztrátové funkce. Právě optimalizátor na základě koeficientu učení (viz 4.2.2) a ztrátové funkce určuje, jak moc a které parametry sítě upravit tak, aby se hodnota ztrátové funkce zmenšila. Problém, se kterým se u optimalizace můžeme setkat, je že i v případě, že budeme provádět optimalizaci nekonečně dlouho, tak se dostaneme pouze do lokálního minima ztrátové funkce, nikoliv globálního. Optimalizátorů existuje několik a jejich volba závisí především na konkrétním případu užití.

Gradientní sestup

Gradientní sestup (*gradient descent*) je užívaný napříč širokou škálou matematických problémů. Důvodem je hlavně rychlost tohoto algoritmu, jeho robustnost a flexibilita. Gradient

je vektor, jehož složky tvoří parciální derivace funkce podle jednotlivých vstupů, který nám určuje, jak by změna ve váhovém ohodnocení nebo jiném parametru ovlivnila hodnotu ztrátové funkce. Princip gradientního sestupu je zachycen na Obrázku 3.2, kde J je ztrátová funkce a $J_{\min}(w)$ je globální minimum ztrátové funkce.



Obrázek 3.2: Princip gradientního sestupu. Převzato z [23].

Jak moc a v jakém směru jsou jednotlivé parametry upraveny se určuje na základě kroku v opačném směru než určuje gradient ztrátové funkce, zapsáno:

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j}, \tag{3.7}$$

kde η je koeficient učení (viz 4.2.2), w_j je j-tá složka vektoru parametrů w a Δw_j značí změnu jednoho parametru. Pokud by jako ztrátová funkce byla využita metrika MSE (viz Rovnice 3.5), tak lze vztah pro výpočet změny jednotlivých parametrů zapsat:

$$\Delta w_j = \eta \frac{1}{N} \sum_{u,i} 2x_j(R_{u,i} - e_{u,i}) \tag{3.8}$$

Parametry jsou poté upraveny po každém průchodu datovou sadou následovně:

$$\vec{w} = \vec{w} + \Delta\vec{w} \tag{3.9}$$

Namísto výpočtu všech gradientů až po průchodu celou datovou sadou je možné sadu rozdělit na dávky (*batches*) a výpočty provádět po těchto dávkách. Tato varianta gradientního sestupu se nazývá **stochastický gradientní sestup** (*stochastic gradient descent*).

Bez velikosti dávky by se parametry sítě upravovaly pouze po průchodu celou datovou sadou. Stanovením velikosti dávky tedy umožníme neuronové síti, aby parametry upravovala častěji. Je rovněž důležité vzít v úvahu, že upravováním vah před průchodem celou datovou sadou vytváříme v datech šum, který ale zpravidla na proces trénování nemá negativní efekt.

Velikost dávky do určitého bodu také souvisí s rychlostí zpracování vzorků a tudíž délkou procesu trénování. Většina knihoven zprostředkovávajících lineární algebraické operace, které jsou používány při trénování, totiž využívá vektorizace za účelem urychlení vektorových a maticových operací. Tím se však navyšují nároky na paměť systému, a jelikož při

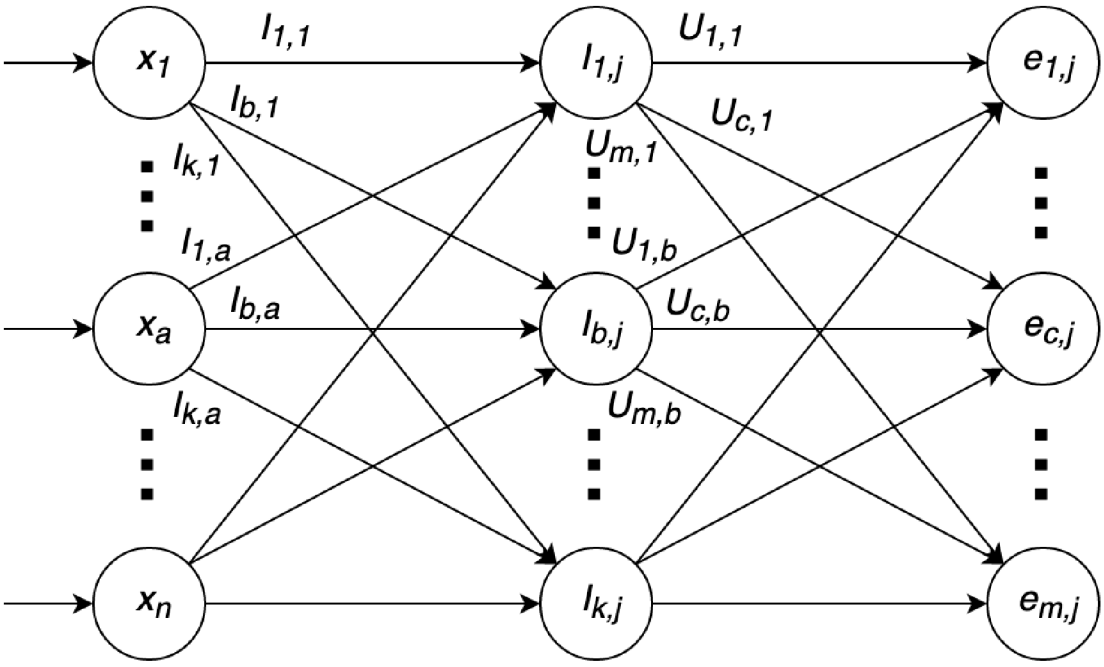
rozdělení datové sady na dávky není v paměti zapotřebí udržovat celou datovou sadu, ale pouze její část, jsou výpočetní operace zpracovány rychleji.

Adam

Adam (*adaptive moment estimation*) je optimalizační algoritmus založený na gradientním sestupu, ale na rozdíl od něj nevyužívá pouze gradient z aktuálního kroku. Tento algoritmus používám v rámci implementovaného faktorizačního modelu, jelikož s jeho využitím dosahuje model nejlepších výsledků. Detailním popisem tohoto algoritmu se v práci zabývat nebudu. Více informací je dostupných v článku [17].

3.2 Využití u doporučovacích systémů

Neuronové sítě je možné u doporučovacích systémů využít různými způsoby. V práci se budu zaměřovat na jejich využití pro kolaborativní filtrování pomocí faktorizace matic (viz 2.1.2). Při faktorizaci matice hodnocení R na matici uživatelů U a matici položek I totiž můžeme vektor charakteristik (*embeddings*) uživatele U_i a vektor charakteristik položky I_j reprezentovat jako vrstvy neuronové sítě. Výstupem sítě je pak vektor předpovězených hodnocení všech uživatelů pro položku j. Pro specifikaci konkrétní položky, pro kterou chceme předpověď provést, je možné položku předat na vstup sítě formou *one-hot* reprezentace, o které se lze dočíst více v článku [3].



Obrázek 3.3: Dopředná neuronová síť pro kolaborativní filtrování pomocí faktorizace matic. U je matice uživatelů a U_i je vektor charakteristik uživatele. I je matice položek a I_j je vektor charakteristik položky.

Příklad takové sítě je ilustrován na Obrázku 3.3, kde m je celkový počet uživatelů, n celkový počet položek a k je dimenzionalita vektorů charakteristik uživatelů a položek.

Jedná se o **dopřednou síť**, přičemž na vstup sítě přichází *one-hot* vektorová reprezentace položky x_j, kde j-tá dimenze tohoto vektoru bude 1 a ostatní 0. Výstup této sítě, tedy vektor předpovědí hodnocení e_j položky j všemi uživateli by pak bylo možné zapsat následovně:

$$e_j = U(Ix_j) \tag{3.10}$$

Pro získání předpovědi hodnocení položky $e_{i,j}$ uživatelem i vynásobíme vektor předpovědí pro položku *one-hot* vektorovou reprezentací uživatele x_i, kde i-tá dimenze tohoto vektoru bude 1 a ostatní 0, zapsáno:

$$e_{i,j} = x_i^T e_j \tag{3.11}$$

Odhad hodnocení $e_{i,j}$ se poté porovná se skutečným hodnocením $R_{i,j}$ a v závislosti na velikosti chyby pak optimalizátor upraví parametry sítě (složky vektorů U_i a I_j) tak, aby se odchylka odhadu hodnocení oproti skutečnému hodnocení zmenšila.

Kapitola 4

Návrh a implementace systému pro doporučování filmů

V této kapitole se lze dočíst o návrhu a implementaci systému, se kterým souvisí volba konkrétní doporučovací techniky, výběr vhodné datové sady pro účely trénování modelu a volba vhodné knihovny, která tento model implementuje.

4.1 Faktorizační model

V době psaní této práce jsou podle [4] téměř všechny doporučovací systémy, které doporučují filmy, z určité části založeny na kolaborativním filtrování. Důvodem je vysoká efektivita těchto systémů a poměrně vysoká přesnost doporučování. Pro kolaborativní filtrování existuje více přístupů zmíněných v sekci 2.1. Přístupy založené na paměti jsou oproti přístupům založených na modelech sice zpravidla jednodušší na implementaci, ale podle [24] dosahují horších výsledků. Rozhodl jsem se proto využít faktorizační model, o jehož principu se lze dočíst v sekci 2.1.2.

Neuronová síť, na které je model doporučovacího systému implementovaný v rámci této práce postaven, může být poměrně jednoduchá, a to proto, že data, na kterých je síť trénovaná, nejsou komplexní. Jedná se pouze o trojice složených z uživatelů, položek a hodnocení. Využitá neuronová síť je popsána v sekci 3.2.

Faktorizační model je již implementovaný v rámci několika volně dostupných knihoven, přičemž v práci se budu zaměřovat na ty, který byly při tvorbě doporučovacího systému využity. Jako ztrátovou funkci jsem zvolil MSE (viz 3.5) a pro optimalizátor jsem zvolil *Adam* (viz 3.1.2).

4.1.1 Open-source knihovny

Poté, co byl zvolen model, na kterém bude doporučovací systém založen, byla provedena analýza několika knihoven, které systémem požadovanou funkcionalitu již částečně implementují. Některé z knihoven obsahují spíše kostru pro tvorbu modelů, zatímco ostatní už v sobě tyto modely mají zcela naimplementovány.

Torch

Torch[1] je knihovna pro strojové učení napsána objektově v jazyce C++. Pro zjednodušení úprav již existujících algoritmů a ulehčení tvorby nových algoritmů a metod byla při návrhu zvolena modulární strategie a logika byla dle [6] rozdělena do následujících tříd:

- `DataSet` – stará se o práci s daty, ať už statickými nebo dynamickými
- `Machine` – slouží například pro reprezentaci neuronových sítí, kdy na základě vstupu a parametrů poskytne výstup
- `Trainer` – používá se pro ladění parametrů třídy `Machine` na základě daných kritérií a datové sady, kterou poskytuje třída `DataSet`
- `Measurer` – slouží pro evaluaci neuronové sítě třídy `Machine`

Nejprve se tedy pomocí třídy `DataSet` vytvoří několik trénovacích příkladů, které se předají na vstup třídy `Machine`, která představuje neuronovou síť. Výstup třídy `Machine` je poté využit třídou `Trainer` pro ladění parametrů neuronové sítě.

PyTorch

PyTorch[2] je vědecký výpočetní balíček založený na *Torch*, který navíc pro urychlení výpočtů umožňuje využití grafických karet. Je to také jedna z preferovaných platforem pro hluboké učení stavěných pro podporu maximální flexibility a rychlosti [26]. Knihovna je navržena tak, aby byla intuitivní a jednoduchá pro adaptaci. Protože se jedná o nativní *Python* balíček, tak se velmi jednoduše zahrnuje do jiných balíčků, modulů, nebo již existujících aplikací. U implementovaného doporučovacího systému tuto knihovnu využívám pro serializaci modelu, aby nebylo nutné při každém spuštění aplikace síť natrénovat znovu.

Spotlight

Spotlight[3] poskytuje již naimplementované reprezentace doporučovacích modelů pro kolaborativní filtrování nebo doporučování založené na sezení. Rovněž obsahuje užitečné nástroje pro získávání i generování datových sad [18].

Tato knihovna je systémem v rámci této práce použita pro implementaci faktorizačního modelu. Zdrojový kód pro tuto přejatou implementaci modelu je součástí přílohy této práce, jelikož jej bylo nutné upravit pro účely experimentů. Ve svém jádru knihovna využívá knihovnu *PyTorch* pro reprezentaci jednotlivých vrstev neuronové sítě a pro již naimplementované optimalizátory, které určují, jak upravovat na základě ztrátové funkce parametry sítě.

4.1.2 Zvolená datová sada

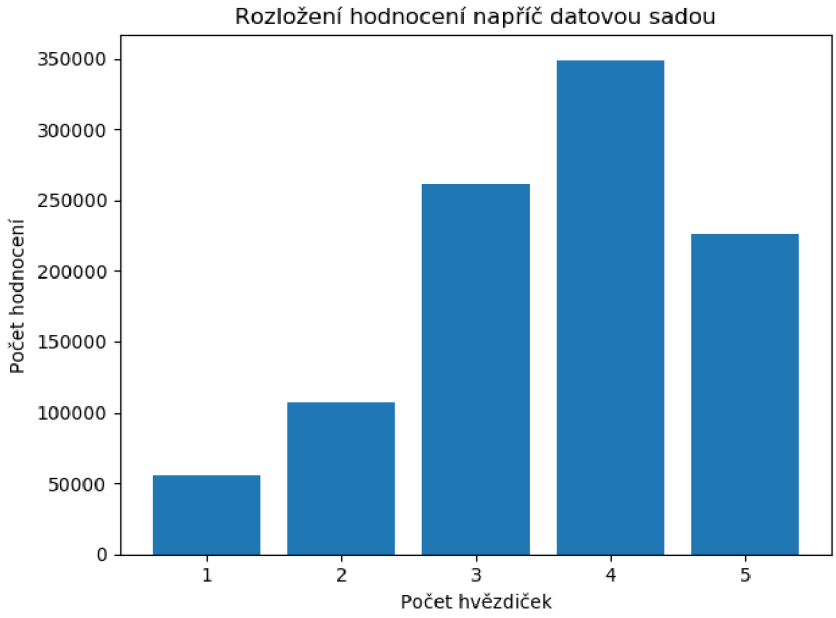
Aby bylo možné experimentovat s doporučovacími technikami, je nutné získat data, která v sobě zahrnují položky, uživatele a reakce uživatelů na některé tyto položky. Reakce se dělí na explicitní, kdy uživatel poskytl zpětnou vazbu například formou hodnocení hvězdičkami,

[1] http://torch.ch/
[2] https://pytorch.org/
[3] https://github.com/maciejkula/spotlight

anebo implicitní, mezi které patří prohlížení položky, čas strávený na stránce, přidání si položky do seznamu přání apod.

Zvolená datová sada pochází z nekomerčního systému pro doporučování filmů zvaného *MovieLens*[4]. Obsahuje **1 000 209 hodnocení**, která udělilo **3 707 filmům 6 041 uživatelů**, spolu s informací o tom, kdy bylo hodnocení provedeno, což je nutné například při trénovaní doporučovacích systémů založených na sezení. Hodnocení nabývá celočíselných hodnot z rozsahu od 1 do 5. Většina datových sad se potýká s problémem nerovnoměrného rozprostření hodnocení, výjimkou není ani tato datová sada, jak lze vypozorovat z Obrázku 4.1. Podle [5] mezi některé z důvodů patří:

- Pokud je film hodnocen záporně, i když například malým počtem hodnocení, tak je nepravděpodobné, že se na něj budou dívat další uživatelé a tudíž záporná hodnocení nebudou přibývat.
- Pokud se film uživateli nelíbil, tak není pravděpodobné, že se bude obtěžovat udělením záporného hodnocení.
- Naopak v případě, že se uživateli film líbil, tak často poskytne kladné hodnocení.



Obrázek 4.1: Rozložení hodnocení napříč *MovieLens* datovou sadou

Důsledky tohoto problému v datových sadách pak po natrénování modelu způsobí, že model převážně doporučuje filmy, které jsou populární. Uživatelům pak nejsou navrhovány filmy například nepopulárních žánrů nebo s kontroverzní tématikou, i když by se jim mohly líbit, jelikož na ně nepřipadá dostatečný počet hodnocení. Tento konkrétní problém lze do určité míry řešit například zavedením penalizace filmů s vysokým počtem hodnocení.

Tuto datovou sadu jsem se rozhodl použít při trénování modelu primárně proto, že se s ní i v současné době provádí mnoho experimentů a to umožňuje provést objektivní

[4] https://movielens.org

posouzení přesnosti mnou implementovaného modelu v porovnání s ostatními řešeními. O tomto srovnání se lze dočíst více v kapitole 6.

4.2 Ladění hyperparametrů modelu

V této sekci jsou rozebrány hlavní konfigurovatelné hyperparametry a ukázán vývoj přesnosti v závislosti na jejich úpravách při využití faktorizačního modelu z knihovny *Spotlight*.

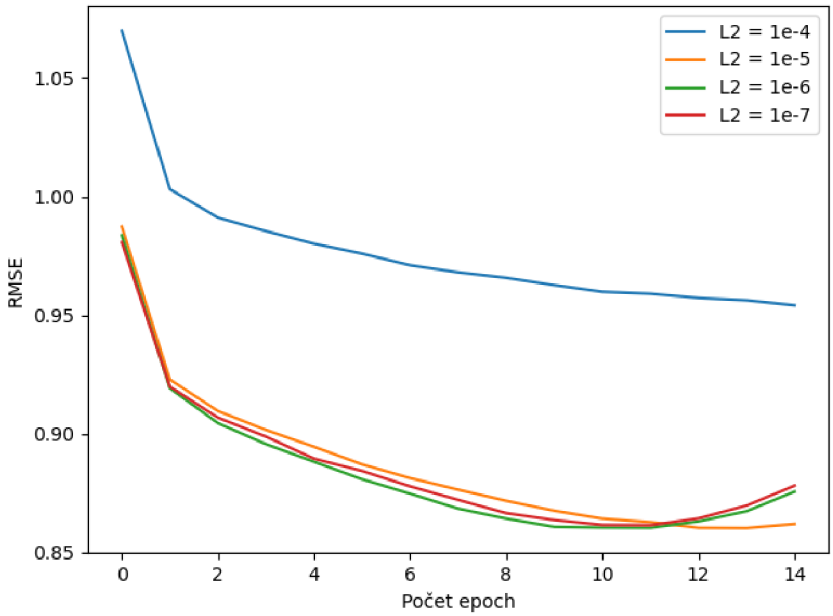
4.2.1 Míra regularizace

Regularizace je technika, jejímž účelem je zabraňovat přetrénování modelu tak, že penalizuje ztrátovou funkci. Regularizačních technik existuje více, přičemž v rámci této práce jsem se setkal s L2 regularizací. Rovnice 3.8 pro výpočet změny jednotlivých parametrů sítě se při využití L2 regularizace upraví takto:

$$\Delta w_j = \eta \frac{1}{N} \sum_{u,i} 2x_j(R_{u,i} - e_{u,i}) + \lambda w_j, \tag{4.1}$$

kde λ je regularizační koeficient. Pokud je zvolená míra regularizace s ohledem na datovou sadu a další parametry příliš nízká, tak může dojít k velmi brzkému přetrénování modelu a tím model dosahuje výrazně horších výsledků. Na druhou stranu při volbě příliš vysoké hodnoty se k ideálnímu nastavení parametrů modelu také nemusíme přiblížit.

Vhodná hodnota tohoto parametru se zpravidla odvíjí od počtu prvků v datové sadě a počtu epoch. V případě, že datová sada obsahuje málo dat a nebo je zvolen menší počet průchodů, tak není riziko přetrénování tak vysoké, a pro tento parametr lze zvolit nižší hodnotu. V opačném případě zase riziko přetrénování narůstá, a proto je při pozorování negativní změny v metrikách přesnosti doporučení zapotřebí zvolit vyšší hodnotu. [16]

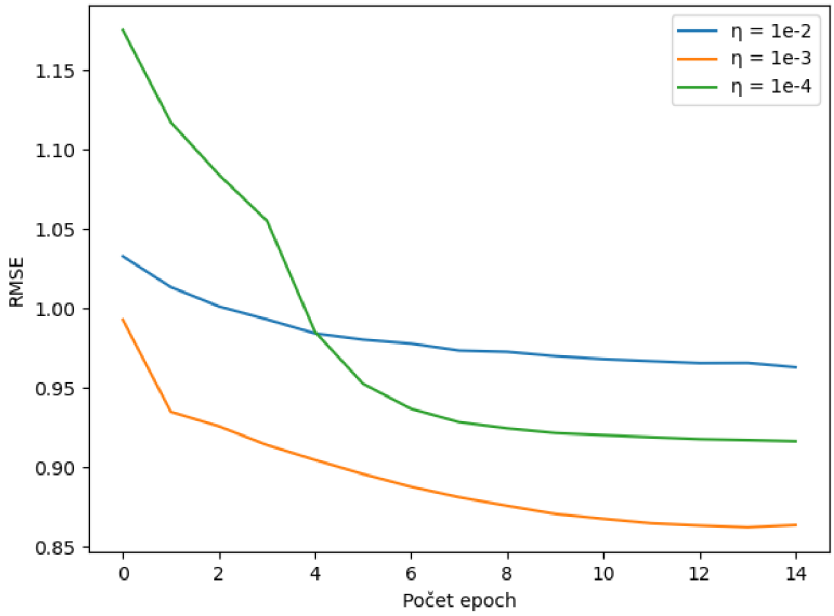


Obrázek 4.2: Odchylka RMSE u validační sady v závislosti na L2 regularizaci.

Na Obrázku 4.2 lze pozorovat vývoj přesnosti na validační sadě s vybranými hodnotami pro L2 regularizaci. Nejlepší výsledky model dosahoval pro hodnotu L2 regularizace $1e-5$. Avšak stejně jako tomu je i u většiny ostatních hyperparametrů, konečná volba hodnoty závisí především na případu užití modelu.

4.2.2 Koeficient učení

Další konfigurovatelný parametr při trénování modelu je koeficient učení (*learning rate*), který nám umožňuje ovlivnit, jak moc jsou při jednotlivých průchodech datovou sadou (nebo dávkách) upravovány parametry sítě.



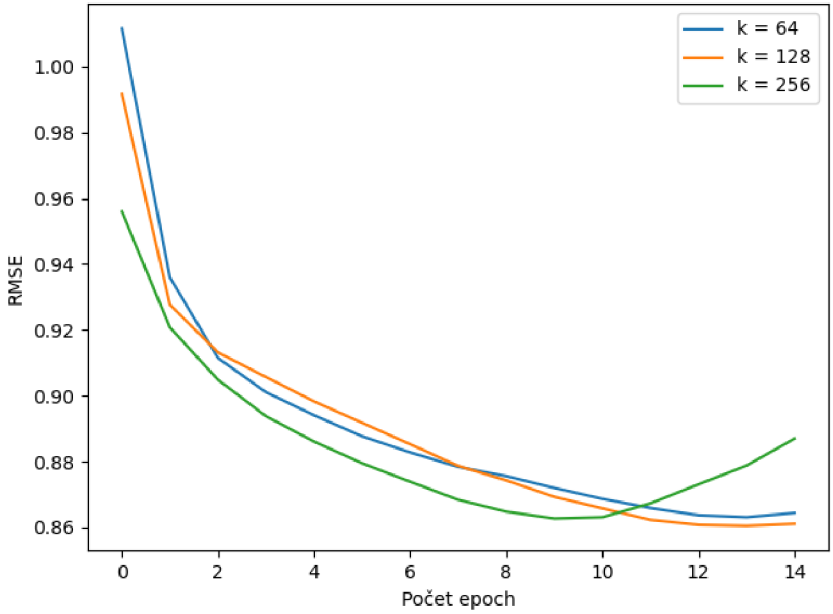
Obrázek 4.3: Odchylka RMSE u validační sady v závislosti na koeficientu učení.

V případě, že pro tento hyperparametr zvolíme příliš vysokou hodnotu, tak se model ani nemusí přiblížit k vhodnému nastavení vah, protože úpravy parametrů sítě v jednotlivých krocích budou příliš markantní. Nízká hodnota konverguje k správnému řešení, ale to pouze s dostatečně velkým počtem průchodů. Pokud by totiž byl počet průchodů s ohledem na zvolenou hodnotu koeficientu učení zanedbatelný, tak se parametry modelu od počátečního nastavení změní pouze minimálně, což také mít za následek špatnou přesnost modelu při doporučování. Vývoj přesnosti na validační sadě pro vybrané hodnoty koeficientu učení η je zachycen na Obrázku 4.3. Nejlepší výsledky model dosahoval pro hodnotu $\eta = 1e-3$.

4.2.3 Dimenzionalita vektorů uživatelů a filmů

Význam dimenzí vektorů charakteristik uživatelů a filmů je před námi skryt. Obecně lze říci, že čím více dimenzionální tyto vektory jsou, tím detailnější vzory (*patterns*) mohou být zachyceny v parametrech sítě. V případě, kdy by byla dimenzionalita vektorů příliš malá, tak model zase nemá k dispozici dostatečný počet parametrů pro reprezentaci těchto vzorů v datech a doporučení budou spíše náhodné [20].

Na Obrázku 4.4 si lze všimnout odchylky na validační sadě pro zvolené hodnoty počtu dimenzí k. Odchylka po 15 průchodech datovou sadou byla nejmenší pro dimenzionalitu $k = 128$.



Obrázek 4.4: Odchylka RMSE u validační sady v závislosti na počtu dimenzí vektorů uživatelů a filmů.

Kapitola 5

Experimenty a jejich vliv na kvalitu doporučení

Tato kapitola obsahuje informace o provedených experimentech, jejich účelu a vlivu na doporučování filmů. V první části se zaměřuje na experimenty s hyperparametry modelu. Dále je zde proveden experiment se zvolenou datovou sadou. Motivací těchto experimentů je především další zlepšení odchylky při doporučování.

5.1 Klesající koeficient učení

Pro první experiment jsem se rozhodl zaměřit na hyperparametr koeficient učení a vliv jeho úprav na výslednou přesnost modelu. Namísto využití konstantního koeficientu učení se po každém průchodu datovou sadou koeficient učení η pro budoucí epochu zmenší vynásobením koeficientem r, zapsáno:

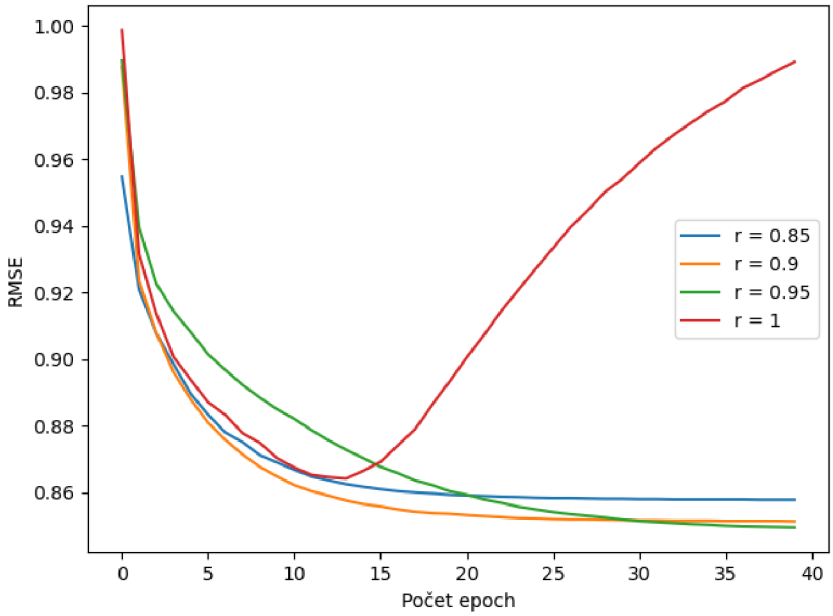
$$\eta_{(t+1)} = r\eta_t \tag{5.1}$$

Počáteční hodnotu pro koeficient učení η_0 jsem na základě výsledků v sekci 4.2 nastavil na $1e-3$. Z Obrázku 5.1 lze vypozorovat, že při stanovení koeficientu r na hodnotu 0.95 se oproti konstantnímu koeficientu učení zlepšila odchylka RMSE o přibližně 0.01, jelikož klesající hodnota koeficientu učení oddálila přetrénování tím, že při posledních epochách optimalizátor upravoval parametry sítě jen minimálně.

5.2 Úprava rozložení hodnocení v datové sadě

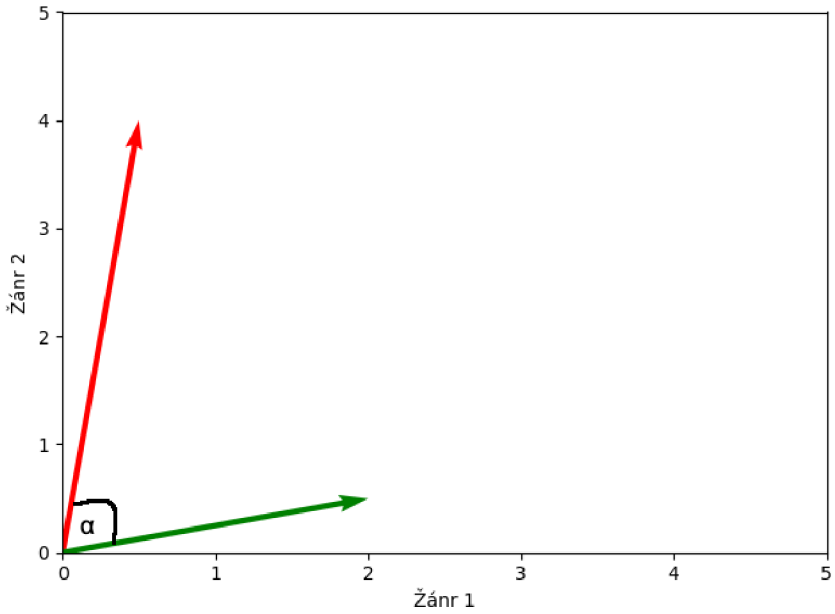
Pro další experiment jsem vybral rozložení hodnocení v datové sadě. Jelikož jsou filmy i uživatelé v modelu reprezentovány formou k-dimenzionálních vektorů, předpověď hodnocení je vždy výsledkem skalárního součinu těchto vektorů. Pro účely tohoto experimentu řekněme, že jsou uživatelé i filmy 2D vektory a jejich dimenze představují žánry, podobně jako v sekci 2.1.2.

Protože jsou hodnocení v *MovieLens* datové sadě dostupné pouze v rozsahu od 1 do 5, tak je vektorový prostor uživatelů a filmů omezen tím, že jejich skalární součin musí také být v rozsahu od 1 do 5. Je tedy žádoucí změnit rozložení hodnocení v datové sadě z toho dosavadního na rozsah od -2 do 2, což umožní modelu ve fázi trénování pracovat s obsáhlejším rozsahem možných hodnot pro jednotlivé dimenze vektorů filmů a uživatelů.

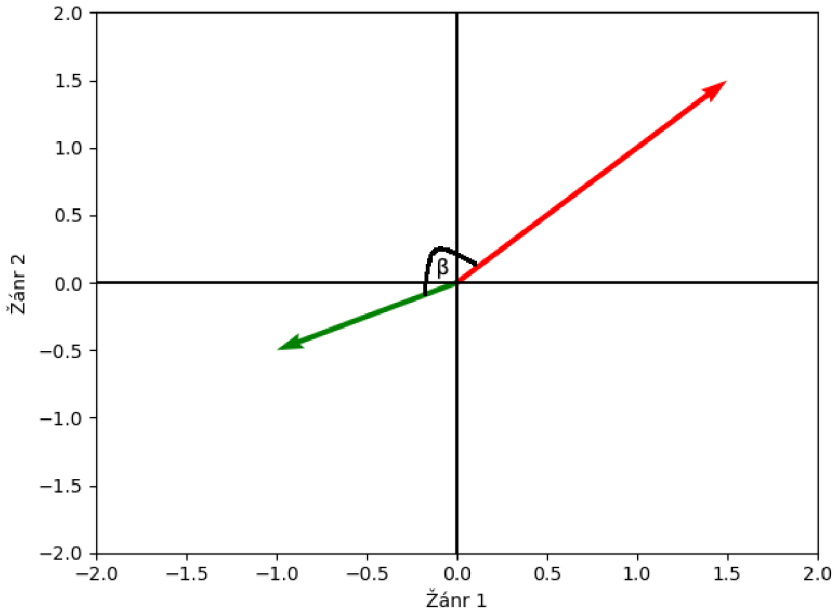


Obrázek 5.1: Odchylka přesnosti RMSE na validační sadě s jednotlivými hodnotami pro koeficient r.

Na Obrázku 5.3 je ilustrován rozdíl při využití jednotlivých rozsahů. Pomocí získaných dat zobrazených na Obrázku 5.3 je možné ověřit, že tento experiment zlepšil odchylku RMSE o přibližně 0.007.

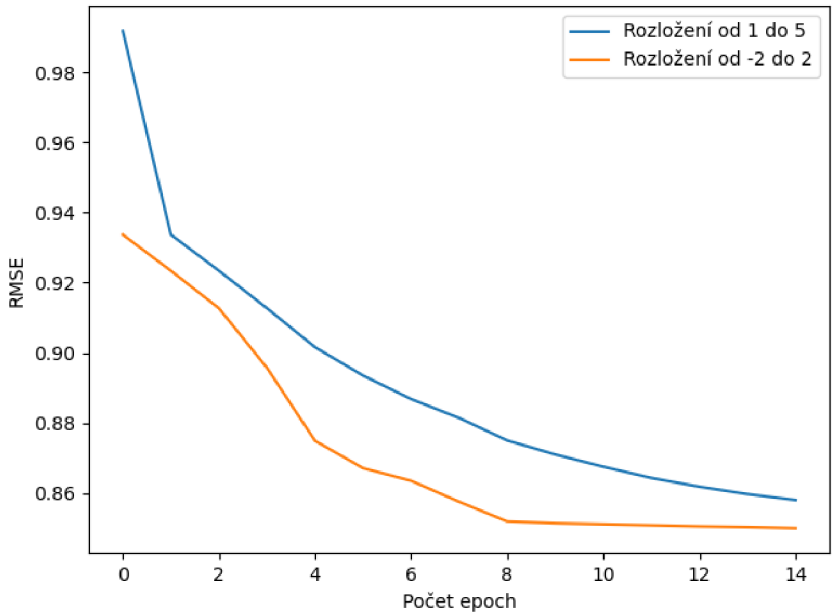


(a) Rozložení datové sady od 1 do 5, kde úhel α může dosahovat maximálně 90°.



(b) Rozložení datové sady od -2 do 2, kde úhel β může dosahovat maximálně 180°.

Obrázek 5.2: Ilustrace rozdílu velikosti vektorového prostoru při rozdělení datové sady.



Obrázek 5.3: Odchylka přesnosti RMSE na validační sadě s rozložením hodnocení od 1 do 5 a poté od -2 do 2.

Kapitola 6

Zhodnocení a porovnání s již existujícími řešeními

V této kapitole se nachází zhodnocení výsledků modelu implementovaného v rámci této práce a jeho srovnání s ostatními modely natrénovanými na stejné datové sadě.

6.1 Dosažené výsledky

Po volbě doporučovací techniky a implementaci této techniky pomocí volně dostupné knihovny jsem se pokusil nalézt ideální kombinaci hyperparametrů pro dosažení co nejvyšší přesnosti při doporučování (viz 4.2). Následně jsem začal provádět různé experimenty za účelem dalšího zlepšení přesnosti (viz 5), které měly na konečnou přesnost modelu kladný efekt.

Průběh trénování výsledného modelu je formou metriky RMSE u jak trénovací, tak testovací sady pro 40 epoch zachycen na Obrázku 6.1. Výsledná odchylka **RMSE testovací sady je u serializovaného modelu rovna** 0.8490 a **MAE dosahuje hodnoty** 0.6676**.** Tyto výsledné metriky jsou použity při porovnávání s ostatními řešeními v sekci 6.2.

6.2 Srovnání přesnosti doporučení

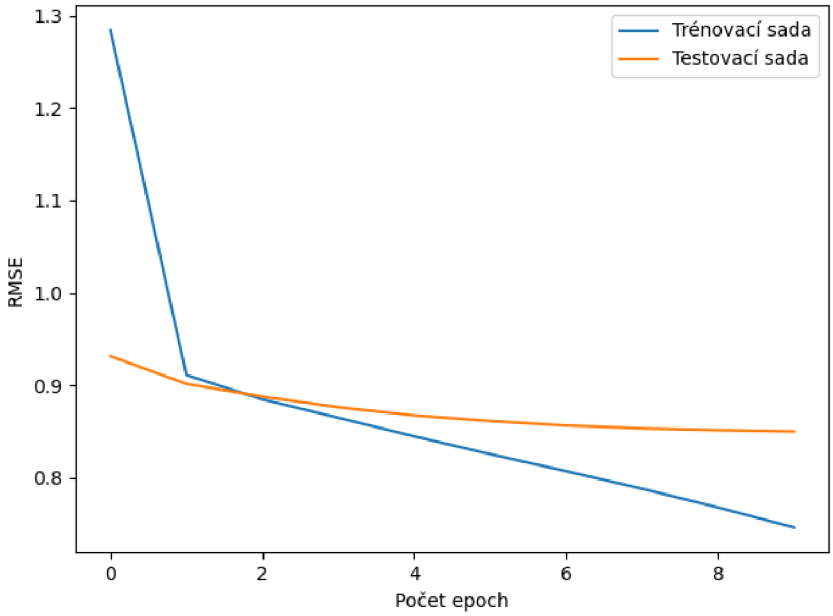
Po zaznamenání dosažených výsledků jsem hledal jiná řešení, která využívala stejnou datovou sadu, aby bylo možné provést objektivní srovnání přesnosti modelu.

6.2.1 Kaggle

Jelikož je datová sada, která je používána při trénování modelu v této práci, volně přístupná na stránce *Kaggle*[1], tak je možné provést objektivní srovnání přesnosti doporučení modelu s jinými řešeními na této stránce.

Uživatel *jftavares*[2] sestrojil doporučovací systém nad datovou sadou *MovieLens* založený na přístupu kolaborativního filtrování mezi uživateli za pomocí algoritmu k-nejbližších sousedů, o kterém se lze dočíst více v článku [25]. Využil k tomu knihovny *Surprise*[3]. Model

[1] https://www.kaggle.com/
[2] https://www.kaggle.com/jftavares/movielens-1m-with-scikit-surprise
[3] http://surpriselib.com/



Obrázek 6.1: Odchylka RMSE u trénovací a testovací datové sady výsledného modelu.

implementovaný tímto uživatelem dosahoval přesnosti RMSE 0.8923 a MAE 0.7012, tedy v porovnáním s mnou natrénovaným modelem horších výsledků.

Pro další porovnání jsem zvolil doporučovací systém, který vytvořil uživatel *indralin*[4]. Využil již připravenou implementaci techniky obsaženou opět v knihovně *Surprise*.

Na rozdíl od řešení předchozího uživatele se rozhodl využít přístup založený na modelech namísto paměťového přístupu. Rovněž kromě hodnocení využil při trénování modelu také metadata obsažená v datové sadě, jako například věk a pohlaví uživatelů, či žánry filmů a léta jejich vydání. Tato úprava se evidentně osvědčila, protože po natrénování dosahoval model přesnost RMSE 0.6991 a MAE zase 0.5412, což je lepší, než model implementovaný v rámci této práce.

6.2.2 Papers With Code

Na stránce *Papers With Code*[5] lze nalézt výsledky existujících řešení mimo jiné v podobě odchylky RMSE.

[4] https://www.kaggle.com/indralin/movielens-project-1-2-collaborative-filtering
[5] https://paperswithcode.com/sota/collaborative-filtering-on-movielens-1m

Tabulka 6.1: Odchylka RMSE na testovací sadě pro 10 nejlepších řešení na *Papers With Code*.

Pořadí	Název	RMSE	Rok
1.	Sparse FC	0.824	2018
2.	CF-NADE	0.829	2016
3.	I-AutoRec	0.831	2015
4.	GC-MC	0.832	2017
5.	I-CFN	0.8321	2016
6.	NNMF	0.843	2015
7.	IGMC	0.857	2019
8.	U-CFN	0.8574	2016
9.	Factorized EAE	0.860	2018
10.	Factorization with dictionary learning	0.866	2016

Podle výsledků zachycených v Tabulce 6.1 mnou implementovaný model dosahuje průměrné přesnosti v porovnání s ostatními řešeními prezentovanými na této stránce.

Kapitola 7

Závěr

V této práci je popsáno množství technik, pomocí nichž je možné sestrojit funkční doporučovací systém. Patří mezi ně statistické metody, ale i složitější přístupy, které využívají strojové učení s pomocí neuronových sítí.

Cílem práce bylo navrhnout a implementovat doporučovací systém založený na technice kolaborativního filtrování a provádět experimenty nad tímto systémem za účelem dosažení co největší přesnosti při doporučování. Pro implementaci systému jsem zvolil faktorizační model založený na neuronových sítích. Ladění hyperparametrů tohoto modelu a provedené experimenty měly na výsledky kladný vliv a proto je implementovaný systém schopný předpovědět hodnocení, které uživatel filmu udělí, s odchylkou RMSE 0.8490. Tato přesnost je v porovnání s již existujícími řešeními pracujícími se stejnou datovou sadou průměrná.

V případě budoucího vývoje systému by bylo zajímavé experimentovat s jinými doporučovacími technikami. Možné by bylo také dále experimentovat s hyperparametry modelu nebo odlišnými datovými sadami a analyzovat důsledky změny přesnosti při stejném nastavení hyperparametrů, jako například vliv rozložení udělených hodnocení v datové sadě. V neposlední řadě by bylo možné například využít superpočítač a zvolit takovou konfiguraci hyperparametrů, která by na běžném stroji způsobila velmi dlouhou dobu trénování, ale pomohla by k dalšímu zlepšení přesnosti.

Literatura

[1] Amatriain, X. a Basilico, J. Recommender systems in industry: A netflix case study. In: *Recommender systems handbook*. Springer, 2015, s. 385–419.

[2] Boyle, E. *Understanding Latent Style*. [Online; navštíveno 12.7.2020]. Dostupné z: https://multithreaded.stitchfix.com/blog/2018/06/28/latent-style/.

[3] Brownlee, J. *Why One-Hot Encode Data in Machine Learning?* [Online; navštíveno 22.6.2020]. Dostupné z: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/.

[4] Calderon, P. *An Overview of Recommendation Systems*. [Online; navštíveno 12.3.2020]. Dostupné z: http://datameetsmedia.com/an-overview-of-recommendation-systems/.

[5] Chakrabarti, S. et al. Data Mining Curriculum: A proposal. *Intensive Working Group of ACM SIGKDD Curriculum Committee*. 2006, roč. 140.

[6] Collobert, R., Bengio, S. a Mariéthoz, J. *Torch: A modular machine learning software library*. Idiap, 2002.

[7] Covington, P., Adams, J. a Sargin, E. Deep Neural Networks for YouTube Recommendations. In: ACM. *Proceedings of the 10th ACM conference on recommender systems*. 2016, s. 191–198.

[8] Davidson, J. a. L. et al. The YouTube video recommendation system. In: ACM. *Proceedings of the fourth ACM conference on Recommender systems*. 2010, s. 293–296.

[9] Draelos, R. *Best Use of Train/Val/Test Splits, with Tips for Medical Data*. [Online; navštíveno 22.4.2020]. Dostupné z: https://glassboxmedicine.com/2019/09/15/best-use-of-train-val-test-splits-with-tips-for-medical-data/.

[10] Ekstrand, M. D., Riedl, J. T., Konstan, J. A. et al. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*. Now Publishers, Inc. 2011, roč. 4, č. 2.

[11] Fang, Z., Zhang, L. a Chen, K. Hybrid Recommender System Based on Personal Behavior Mining. *ArXiv preprint arXiv:1607.02754*. 2016.

[12] Gomez Uribe, C. A. et al. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information Systems (TMIS)*. ACM. 2016, roč. 6, č. 4, s. 13.

[13] Haykin, S. *Neural networks: a comprehensive foundation.* Prentice Hall PTR, 1994.

[14] Hidasi, B., Karatzoglou et al. Session-based Recommendations with Recurrent Neural Networks. *ArXiv preprint arXiv:1511.06939.* 2015.

[15] Karat, S. *How do you build a "People who bought this also bought that"-style recommendation engine.* [Online; navštíveno 18.12.2019]. Dostupné z: https://datasciencemadesimpler.wordpress.com/2015/12/16/understanding-collaborative-filtering-approach-to-recommendations/.

[16] Khandelwal, R. *L1 and L2 Regularization.* [Online; navštíveno 18.1.2020]. Dostupné z: https://medium.com/datadriveninvestor/l1-l2-regularization-7f1b4fe948f2.

[17] Kingma, D. P. a Ba, J. Adam: A Method for Stochastic Optimization. *ArXiv preprint arXiv:1412.6980.* 2014.

[18] Kula, M. *Spotlight* [https://github.com/maciejkula/spotlight]. GitHub, 2017.

[19] Lops, P., De Gemmis, M. a Semeraro, G. Content-based recommender systems: State of the art and trends. In: *Recommender systems handbook.* Springer, 2011, s. 73–105.

[20] Melville, P. a Sindhwani, V. Recommender Systems. *Encyclopedia of Machine Learning and Data Mining.* Springer. 2017, s. 1056–1066.

[21] Miljanovic. Comparative Analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction. *Indian Journal of Computer Science and Engineering.* Citeseer. 2012, roč. 3, č. 1, s. 180–191.

[22] Mohri, M. et al. *Foundations of Machine Learning.* MIT press, 2018.

[23] Raschka, S. *Gradient Descent and Stochastic Gradient Descent.* [Online; navštíveno 18.3.2020]. Dostupné z: http://rasbt.github.io/mlxtend/user_guide/general_concepts/gradient-optimization/.

[24] Ricci, F. et al. *Recommender systems: introduction and challenges.* Springer, 2015.

[25] Sharma, N. *Recommender Systems with Python - Part II: Collaborative Filtering (K-Nearest Neighbors Algorithm).* [Online; navštíveno 10.4.2020]. Dostupné z: https://heartbeat.fritz.ai/recommender-systems-with-python-part-ii-collaborative-filtering-k-nearest-neighbors-algorithm-c8dcd5fd89b2.

[26] Shetty, S. *What is PyTorch and how does it work?* [Online; navštíveno 2.1.2020]. Dostupné z: https://hub.packtpub.com/what-is-pytorch-and-how-does-it-work/.

[27] Zbořil, F. V. *Slidy k předmětu Soft computing - Přednáška 1.* [Online; navštíveno 22.7.2020]. Dostupné z: https://www.fit.vutbr.cz/study/courses/SFC/private/19sfc_1.pdf.

[28] Zell. *Simulation Neuronaler Netze.* Addison-Wesley Bonn, 1994.

Příloha A

Obsah CD

Na přiloženém CD je možné najít tyto materiály:

- **Zdrojové soubory programu**: `recommender_system/src`
- **Zkompilovaná aplikace**: `recommender_system/build`
- **Zdrojové soubory technické zprávy**: `recommender_system/latex`
- **Zkompilovaná technická zpráva**: `recommender_system/thesis`

Soubor `recommender_system/src/lib/spotlight/factorization/explicit.py` obsahuje upravenou implementaci přejatého modelu z knihovny *Spotlight* (viz 4.1.1). Kvůli experimentům byl totiž nutný zásah do zdrojového kódu knihovny. Upravené řádky jsou řádně označeny.

Příloha B

Manuál

Zkompilovaná aplikace je obsažena ve složce `build`. V této složce se také nachází již předem serializovaný model. Pro znovu natrénování modelu je vyžadováno připojení k internetu pro stažení datové sady.

Spuštění ze zdrojových souborů

Pokud není předem zkompilovaná aplikace spustitelná, je zapotřebí ji spustit ze zdrojových souborů. K tomu je zapotřebí nainstalovat správce balíčku *Conda*[1] pro instalaci závislostí.

Všechny potřebné knihovny lze poté nainstalovat vytvořením nového virtuálního prostředí příkazem `conda env create -f environment.yml` provedeným ze složky se zdrojovým kódem. Je možné, že budou poté pro spuštění chybět některé systémové knihovny, které jsou však odlišné pro jednotlivé systémové instalace a nelze je zde všechny vypsat. Po stažení všech závislostí je nutné aktivovat nové virtuální prostředí příkazem `conda activate` a poté je možné aplikaci spustit příkazem `python3.6 main.py` také ze složky se zdrojovým kódem.

Ovládání

Při spuštění aplikace bez parametrů se nejprve aplikace pokusí v pracovním adresáři najít serializovaný model. Pokud serializovaný model nebude existovat, tak bude natrénován znovu. V případě, že serializovaný model existuje, ale uživatel chce, aby se model natrénoval znovu, je zapotřebí využít odpovídající přepínač `-t`.

Výstupem aplikace je evaluace modelu pomocí metrik MAE a RMSE. Ve zdrojovém kódu je fixně nastaven *seed*, který zajišťuje stejný výsledek i při znovu natrénování modelu.

[1] https://docs.conda.io/en/latest/