Doporučovací systém

Jan Tilgner, Ondrej Šajdík, Dan Plaček

29. května 2021

1 Zadání

V doporučovacím systému řešíte úlohu, kde máte velké množství uživatelů, velké množství produktů a řídkou "matici"hodnocení. Na základě toho se se snažíte "dopočítat"chybějící hodnocení. Rozumné přístupy vytvoří embedding uživatelů a produktů a z toho hodnocení dopočítávají. Zároveň je dobré použít co nejvíc informací, jako uživatelské profily, popisy produktů a texty hodnocení. Přehled dostupných datových sad je zde.

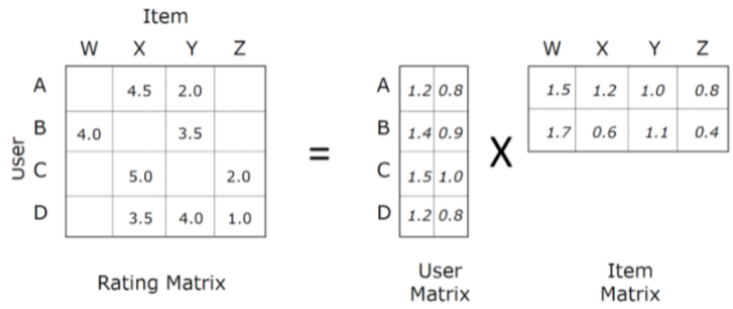
2 Existující řešení

V oblasti doporučovacích systému se používají dva základní přístupy:

1. **Filtrování podle obsahu** používá atributy položek k doporučení dalších položek podobných tomu, co se uživateli líbí, na základě jeho předchozích akcí nebo explicitní zpětné vazby.
2. **Kolaborativní filtrování** doporučuje položky na základě podobnosti chování uživatelů.

2.1 Matrix Factorization

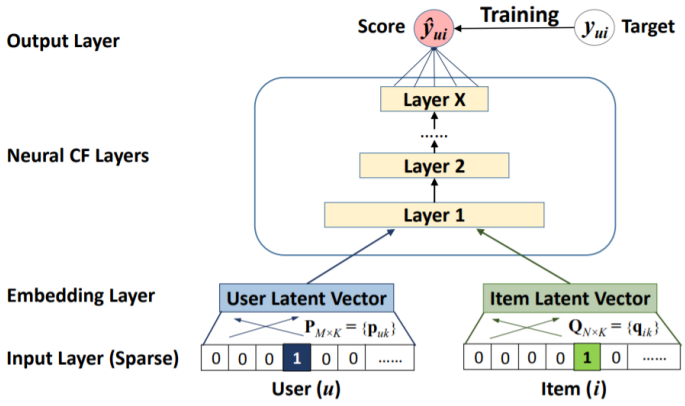
Matrix factorization je jedním z nejpoužívanejších algoritmů pro doporučovací systémy. Využívá již zmíněné kolaborativní filtrování. Na vstup dostane řídkou matici s hodnoceními položek jednotlivými uživateli. Cílem je dopočítat chybějící hodnoty. Algoritmus rozloží matici na produkt dvou matic menší dimenzionality využitím nějakého optimalizačního algoritmu. Např. gradient descentu. Jejich vynásobením dostáváme matici s dopočítanými hodnotami. Viz obrázek 1.



Obrázek 1: Matrix factorization

2.2 Neural Collaborative filtering

Jedná se o kombinaci Matrix factorization a neuronové sítě. V našem baseline řešení byl použit Multilayer Perceptron s věžovou architekturou, kde velikost každé následující vrstvy byla poloviční. Vstup se skládá ze dvou vektorů, kde první identifikuje uživatele a druhý položku. Tyto vektory vznikly jako embedding viz obrázek 2. Tato metoda je popsaná v článku [KM18].



Obrázek 2: Neural collaborative filtering

3 Dataset

Rozhodli jsme se pro použití datasetu od společnosti Steam s daty o uživatelích a hrách z australského serveru. První soubor *australian_users_items.json* obsahuje informace o jednotlivých hráčích a o hrách, které vlastní. Další soubor *steam_games.json* obsahuje metadata her. Hry i uživatelé jsou identifikováni pomocí ID společné pro oba soubory. Data z těchto souborů je tedy možné propojit. V souborech se na každém řádku nachází jeden záznam o hráči či hře. Oba soubory jsou ve tvaru json, ale některé řádky nebyly zcela správně, takže bylo potřeba ošetřit některé chyby při parsovaní souborů.

V prvním souboru se nachází na každém řádku jeden uživatel a seznam jeho her i s daty o tom, jak dlouho hráč danou hru hrál. Z každého záznamu nás zajímá identifikace hráče a jeho her, tedy *user_id* a *game_id*. Tyto informace byly vypsány do .csv souboru, ve tvaru user_id game_id. Každý pár hráč-hra je tak na zvláštním řádku. Pro větší relevanci jsme odfiltrovali hráče, kteří vlastní méně než 5 her. U her to nebylo potřeba, žádná hra se nenacházela u méně než 5 hráčů. Výsledný soubor nám pak slouží jako počáteční matice, která je základem pro učení neuronové sítě.

V druhém souboru se nachází na každém řádku záznam s metadaty o hře. Zajímavé jsou údaje o ceně hry, jejím vydavateli a také žánry a tagy této hry. Jelikož tagy a žánry se u každé hry nachází v nějaké kombinaci vytvořili jsme z nich vektor nul a jedniček, kde 0 znamená, že se tag/žánr u hry nenachází a 1 znamená, že se zde nachází. K uložení těchto informací byl vytvořen druhý csv soubor, který na každém řádku obsahuje záznam o hře, kde je ve sloupcích postupně hodnota *game_id*, *publisher* a *price* a následně vektor nul a jedniček pro jednotlivé tagy a žánry.

Kvůli tomu, že se ID při batchování vzorků generuje z rozsahu 0 až počet vzorků, bylo nutno přidělit hráčům i hrám nová id, aby byl vyplněn tento rozsah. Hry, které měly stejné ID před změnou ho mají stejné i po změně, aby se zachovala reference mezi soubory.

4 Základní řešení

4.1 Model

Pro základní řešení byla převzata architektura NCF popsána výše. Naše konkrétní implementace používá osmi dimenzionální embbedingy uživatelů a položek vložených na vstup sítě. Samotná síť má 2 lineární vrstvy s aktivační funkcí *ReLU* o velikosti 16x64x32x1. A výstupní vrstvu pro vypočteni pravděpodobnosti interakce s aktivační funkcí *Sigmoid*.

4.2 Trénování

Nejprve jsme rozdělili náš dataset na tři části: trénovací, validační a testovací data. Nejprve jsme od každého uživatele vzali jeden záznam a tyto záznamy jsou použity jako testovací data. Ze zbývajících data bylo 90 % dat použito jako trénovací a zbývajících 10 % jako validační. Pro vytváření batchů byla

použita třída DataLoader z knihovny *torch*. Pro trénování jsme používali batchsize 512. Jelikož máme k dispozici jen pozitivní (vlastněné), prvky, přidali jsme za každý takový prvek 4 negativní záznamy o hrách, které uživatel nevlastní. Pozitivní záznamy jsou potom značeny pomocí jedničky, negativní pomocí nuly.

5 Vyhodnocení

Vyhodnocování probíhalo pomocí metrik Hit Ratio a NDCG (Normalized discounted cumulative gain), které byly použity v odborném článku a který používá náš dataset. Obecně u obou metrik vyhodnocování funguje tak, že je zvolen 1 úspěšný produkt uživatele (který již vlastní) a 100 neúspěšných produktů (které nevlastní), ty jsou klasifikovány a tím je vytvořen seznam 101 produktů seřazených od nejpravděpodobnějšího k doporučení po nejméně pravděpodobný. Ten se následně evaluuje pomocí obou metrik a tyto hodnoty se v rámci metrik, přes všechny běhy (uživatele) průměrují.

Hit ratio (HR@k) je vyhodnocovací metrika, která zaznamenává úspěšnost doporučení produktu na základě seřazené řady produktů a to tak, že zjišťuje, zda mezi k nejpravděpodobnějšími produkty je náš úspěšný (vlastněný) produkt. Pokud ano, bereme tento běh jako úspěšný - 1, pokud ne, tak jako neúspěšný - 0. Parametr k byl podle odborného článku zvolen 10.

Metoda Normalized discounted cumulative gain (NDCG@k) funguje na podobném principu, kdy opět vyhodnocuje k produktů s největší pravděpodobností vybrání, ale v tomto případě i ohodnocuje jejich samotné pořadí mezi samotnými k produkty. Tzn. pokud je produkt algoritmem vyhodnocen jako nejpravděpodobnější je ohodnocen 1, ale pokud jako 10. v pořadí, je ohodnocen pouze hodnotou 0,29. Opět podle odborného článku hodnota parametru k byla zvolena jako 10.

Nejdříve se v NDCG spočítá suma relevancí.

$$CG_k = \sum_{i=1}^{k} rel_i$$

Abychom penalizovali horší umístění v rámci k prvků vypočítáme DCG (discounted cumulative gain), kdy výše položené položky z k prvků dosahují vyššího skóre.

$$DCG_k = \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)}$$

Pro spočítání normalizovaného DCG potřebujeme i hodnotu ideálního DCG, které vznikne při ideální klasifikaci (každý klasifikovaný prvek bude na prvním místě).

$$IDCG_k = \sum_{i=1}^{|REL_k|} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

Nakonec vytvoříme normalizovanou hodnotu podělením našeho DCG ideálním DCG.

$$NDCG_k = \frac{DCG_k}{IDCG_k}$$

6 Úpravy

V baseline řešení jsme využívali pouze matici hráčů a jejich her, proto jsme se rozhodli přidat další příznaky, aby se zlepšila výkonnost sítě. Informace o vydavateli byla vložena jako embedding, a zbylé informace (cena, a vektor tagů a žánrů) byly připojeny tak jak jsou. Vstupem sítě tak nakonec je vektor, který vznikl spojením embeddingu hráče, embeddingu hry, embeddingu vydavatelů, ceny a vektoru tagů a žánrů.

6.1 Tvar neuronové sítě

Jelikož se nám změnil počet vstupních příznaků na 87. Velikosti vrstev jsou tedy $87 \times 64 \times 32 \times 1$.

7 Experimenty

7.1 Baseline experimenty

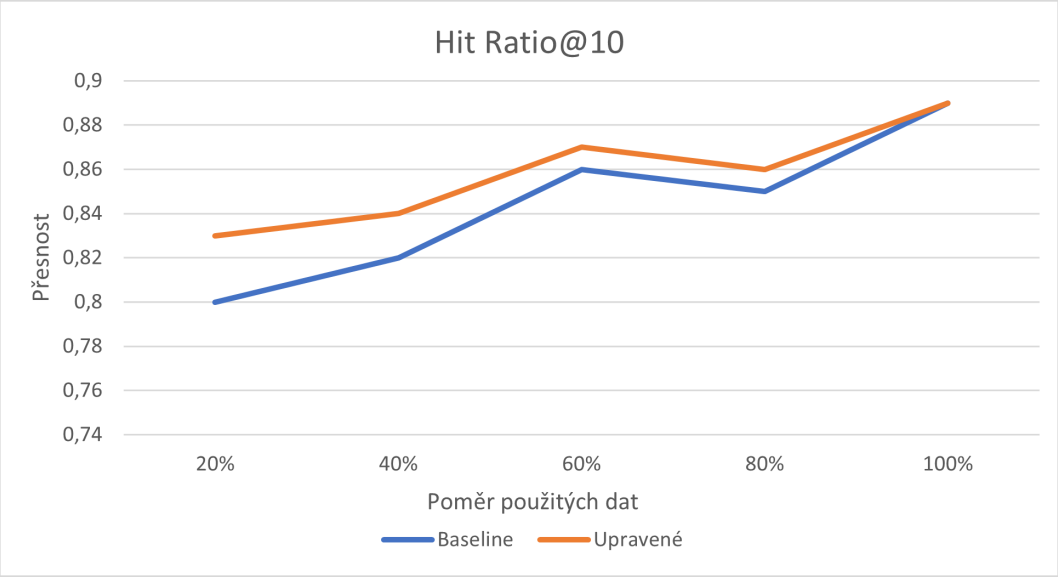
Nejdříve jsme na našem baseline řešení provedli vyhodnocení na původním datasetu a výsledky porovnali s výsledky uvedené v článku [KM18]. Ukázalo se, že přesnost je totožná. Následovné srovnávaní bylo vyhodnoceno na novém datasetu (chybí hry bez metadat).

7.2 Experimenty s vylepšeným řešením

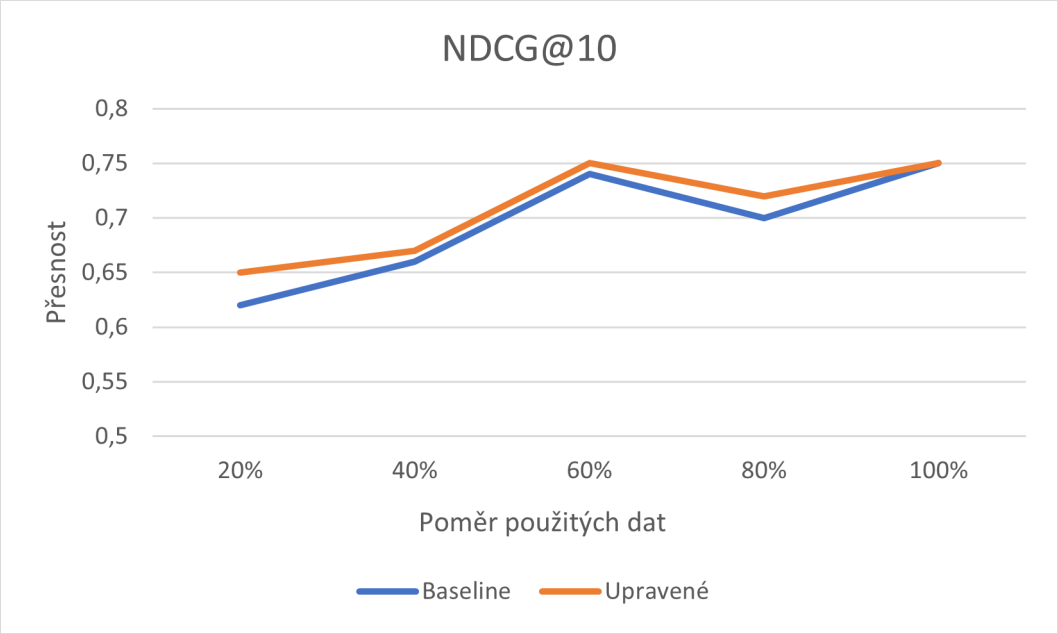
1. **Architektura sítě** - Jelikož se nám změnil počet vstupních příznaků, pokusili jsme se změnit i architekturu sítě, tím, že jsme přidali jednu lineární vrstvu hned na začátek. Zkusili jsme velikosti 64 a 128. Oba pokusy ale vedly ke zhoršení přesnosti obou metrik.
2. **Dimensionalita embeddingu** - Pokoušeli jsme se také změnit dimensionalitu embeddingu. Při dimensionalitě menší než 4 se přesnost snížila. Jinak zůstávala stejná.
3. **Počet negativních vzorků** - Experimentovali jsme i s počtem negativních vzorků na jeden pozitivní. Jako nejlepší nám vyšel poměr 4 ku 1.

8 Výsledky

Porovnání výsledků metriky Hit Ratio@10 je vidět na obrázku 3 a NDCG@10 na obrázku 4. Je vidět, že při použití všech dat se ani jedna metrika téměř vůbec nezlepšila, ale čím menší část datasetu se používá, tím jsou výsledky upravené sítě lepší.



Obrázek 3: Porovnání Hit Ratio@10



Obrázek 4: Porovnání NDCG@10

9 Závěr

Vytvořili jsme baseline řešení, které využívá collaborative filtering založený na vztahu uživatelů a her. K tomuto řešení jsme přidali prvky item-based filteringu v podobě přidání metadat o hrách jako je cena hry, žánr, vydavatel a podobně. Pro takto rozšířený vstup bylo provedeno několik experimentů pro získání nejlepšího nastavení sítě. Výsledné řešení bylo vyhodnocené na různě velkém množství dat a zjistili jsme, že při použití menšího počtu dat se přesnost sítě zlepšuje oproti baseline, ale při použití celého datasetu se tato výhoda vytrácí. Tento projekt můžete nalézt zde [1].

Reference

[KM18] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206. IEEE, 2018.

[1] https://github.com/DanPlacek/Recommender-System