Препорачување филмови со невронски мрежи

Благоја Савевски

3 февруари 2025 год.

Абстракт

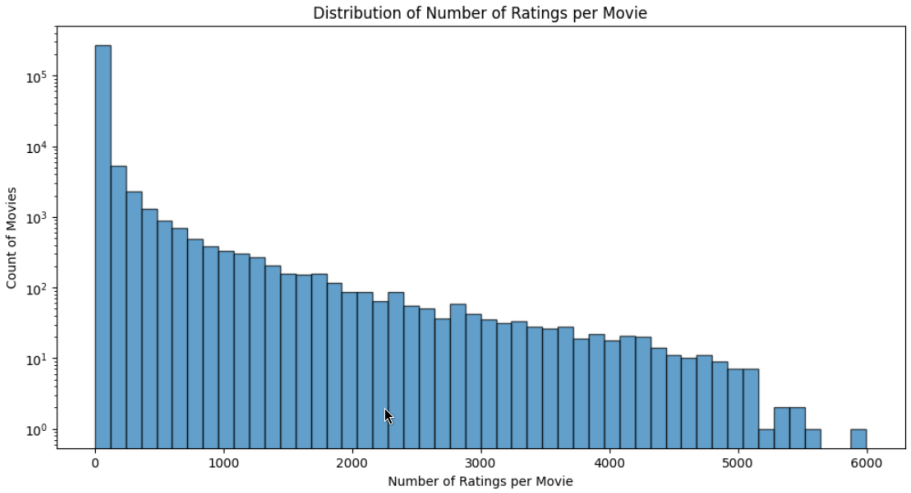
Овој труд ги истражува методите за препорака на филмови со примена на невронски мрежи, со посебен акцент врз комбинираната архитектура NeuMF која ги интегрира техниките на матрична факторизација (GMF) и длабоки мултислојни перцептрони (MLP). Во трудот се презентираат два пристапи за оптимизација: класичниот pointwise модел, кој се тренира со средна квадратна грешка (MSE loss), и pairwise моделот, кој ги максимизира релативните рангирања преку Bayesian Personalized Ranking (BPR loss). Преку експерименти и компаративна евалуација со метрики како Hit Rate@10, NDCG@10 и ROC AUC, се покажува дека, иако моделот со MSE loss обезбедува солидни резултати за општо рангирање на филмовите, моделот со BPR loss значително го подобрува релативното рангирање на препорачаните филмови, што е одлично при услови на имплицитни податоци. Дополнително, трудот ги разгледува техниките за зголемување на информациите, како што се контекстуалните и unsupervised методи, за да се адресира проблемот со "cold start"и да се обезбеди подобра персонализација. Овој труд нуди сеопфатен преглед на современите пристапи во препорачувачките системи и ја демонстрира ефективноста на невронските мрежи во оваа апликација.

1 Вовед

Системите за предложување соодветен производ на корисникот играат клучна улога во подобрувањето на корисничкото искуство во најразлични апликации. Основниот предизвик е да се откријат преференциите на корисникот врз основа на неговите претходни интеракции со системот, што претставува идеална можност за примена на интелигентни агенти, особено оние засновани на длабоки невронски мрежи (DNNs). Овие модели овозможуваат нумеричка апроксимација на карактеристиките на корисникот и множеството на производи, овозможувајќи попрецизно препорачување. Во основата на најпопуларните системи за предложување лежи комбинираната архитектура Generalized Matrix Factorization (GMF) + Multi-Layer Perceptron (MLP), позната како NeuMF. Целта на овој проект е да се истражи нејзиното функционирање, како и да се прошири моделот со интегрирање на дополнителни нумерички и текстуални информации. Целта на ова збогатување на податоците е да се зголеми варијабилноста и адаптивноста на препораките, што би го подобрило генералното функционирање на системот. Конкретната имплементација на моделот е препорачување на филмови врз база на добро познат вебсајт за дискусија и оценување на филмови Letterboxd.

2 Податочно множество

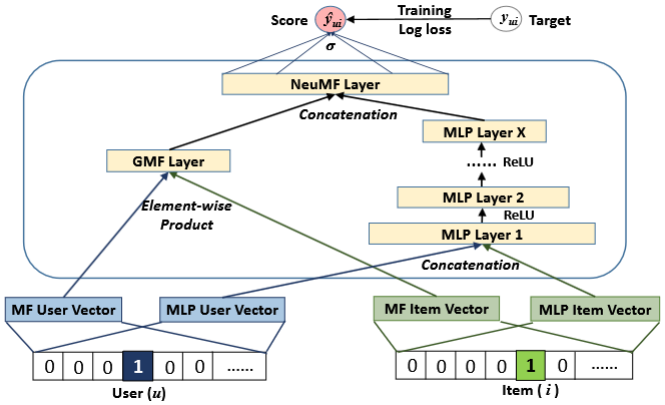
Податочното множество искористено во овој труд е јавно достапно на линкот [1].



(а) Фрекфенција на рејтинзи за секој филм

	Филмови	Корисници	Рејтинзи
n	286069	7477	11078161

(б) Вкупен број на податоци од секој тип во податочното множество. Долу, популарност на филмот vs просечен рејтинг, со боја означен бројот на рејтинзи во ratings табелата



Слика 2: NeuMF архитектура, [2]

3 Neu-MF архитектура

3.1 Matrix Factorization

Првата цел во системот на препорачување е моделирање на преференците на корисникот врз база на претходните негови активности во системот. Најпознатата техника за тоа претставува матрична факторизација - MF, во која корисникот и производот (во овој случај филм) се мапирани во ист повеќедимензионален векторски простор. Интеракцијата на корисникот и филмот се сведува на одредена функција - производ на овие вектори.

$$\hat{y_{ui}} = \boldsymbol{p}_u^T \boldsymbol{q}_i = \sum_{k=1}^{K} = p_{uk} q_{ik} \tag{1}$$

каде $\hat{y_{ui}}$ претставува предвидената вредност од моделот (во случај на конкретниот проблем предвиден рејтинг на филмот).
Се јавува проблем кога рангирањето на еден продукт над друг може транзитивно да влијае на понатамошни рангирања при што би се добило автоматски влошено рангирање.[2] Познато е дека моделите на матрична факторизација трпат overfit како резултат на неопходното зголемување на димензионалноста на латентниот простор како решение на претходноцитираниот проблем, па заедно со MF во NeuMF моделите се воведува и длабока невронска мрежа (Deep Neural Network - DNN).

3.2 DNN - Multi Layer perceptron

Со цел да се овозможи на мрежата да бидат научени интеракциите меѓу информации вметнати во векторите на корисниците и филмовите, се воведува класична Multi Layer Perceptron повеќеслојна мрежа, чијашто активациска функција (неопходна за моделирање на нелинеарни релации) е земена како ReLU, која за скриените (внатрешни) слоеви емпириски е докажана како најдобар избор за невронски мрежи. Заклучно, моделот може да се дефинира како

$$\begin{aligned}
\boldsymbol{y_1} &= \begin{bmatrix} \boldsymbol{p_u} \\ \boldsymbol{q_i} \end{bmatrix}, \\
\boldsymbol{y_2}(y_1) &= a_2(\boldsymbol{W_2}^T \boldsymbol{y_1} + b_2) \\
&\vdots \\
\boldsymbol{y_L}(y_{L-1}) &= a_L(\boldsymbol{W_L}^T \boldsymbol{y_{L-1}} + b_L) \\
\hat{y_{ui}} &= \sigma(\boldsymbol{h}^T \boldsymbol{y_L}(y_{L-1}))
\end{aligned}$$

4 Евалуација

Традиционалната евалуација на модел од невронските мрежи се сведува на добивање предвидувања на моделот на тест множество податоци со кои тој се нема сретнато во фазата на тренинг. Во случајот на податочното множество филмови и соодветни рејтинзи од сајтот Letterboxd, бројот на рејтинзи (повеќе од 11 милиони) е премногу голем за овој метод, па се користи негативно семплирање. Се зема мал број (1-3) филмови кои корисникот ги оценил и значително поголем број филмови со кои корисникот немал интеракција. Моделот работи соодветно ако при рангирањето на сите овие филмови ги фаворизира оние гледани од корисникот во првите k, па се дефинираат метриките Hit Rate@k - просечното „погодување“ на гледаниот филм во првите к, и NDCG@k - мерка која ја прикажува блискоста на произлезеното рангирање со идеалното, дефинирана како:

$$NDCG@k = \frac{DCG@k}{IDCG@k}$$

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

$$IDCG@k = \sum_{i=1}^{k} \frac{2^{rel_{\sigma(i)}} - 1}{log_2(i+1)}$$

каде $\frac{1}{log_2(i+1)}$ ја вреднува повисоко цената за највисоко рангираните елементи, а членот во броителот е 1 за елемент точно рангиран во првите к, 0 инаку. Кодот за имплементација на негативното семплирање на 99 филмови со кои корисникот се нема сретнато и еден кој веќе го има избрано да го рангира изгледа вака:

```
unique_user_ids = ratings_df['user_id'].unique()


k = 10
# Lists for evaluation metrics
hits = []
ndcgs = []
all_true_labels = []
all_pred_scores = []
recommended_items = []
for user_id in tqdm(unique_user_ids, desc="Evaluating users"):
...
hit_rate = np.mean(hits)
ndcg = np.mean(ndcgs)
auc = roc_auc_score(all_true_labels, all_pred_scores)

print(f"Hit Rate@{k}: {hit_rate:.4f}")
print(f"NDCG@{k}: {ndcg:.4f}")
print(f"AUC: {auc:.4f}")
```

```
for user_id in tqdm(unique_user_ids[:100], desc="Processing users"):
    test_items = test.loc[test['userID'] == user_id & (test['prediction']), 'itemID'].values
    pos_items = set(test_items)
    limited_pos_items = list(pos_items)[:3] # take only 3, on processor only takes too slow for more
    user_train_items = train_dict[user_id] | set(limited_pos_items)
    possible_negatives = list(all_items_set - user_train_items)

    for pos_item in limited_pos_items:
        neg_sample = random.sample(list(all_items_set - train_dict[user_id] - {pos_item}), 99)

        item_batch = [pos_item] + neg_sample
        user_batch = [user_id] * len(item_batch)

        scores = ncf_model.predict([np.array(user_batch), np.array(item_batch)])
        scores = np.squeeze(scores)

        labels = np.array([1] + [0]*99) # label the highest rated item as positive
        all_true_labels.extend(labels)
        all_pred_scores.extend(scores)

        sorted_indices = np.argsort(scores)[::-1]
        sorted_labels = labels[sorted_indices]
        sorted_items = np.array(item_batch)[sorted_indices]

        top_k_items = sorted_items[:k]
        recommended_items.extend(top_k_items.tolist())

        if 1 in sorted_labels[:k]:
            hits.append(1)
        else:
            hits.append(0)

        rank = np.where(sorted_labels[:k] == 1)[0]
        if len(rank) > 0:
            ndcgs.append(1 / np.log2(rank[0] + 2))
        else:
            ndcgs.append(0)
```

4.1 Избор на loss функција

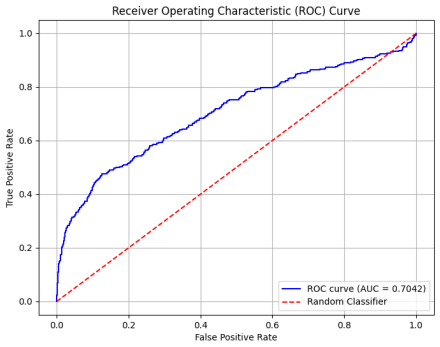
Карактеристично за задачата е тоа што истовремено се работи со имплицитни и експлицитни податоци. Имплицитните податоци се однесуваат на фактот дека корисникот воопшто гледал одреден филм, што може да се интерпретира како индиректна индикација за интерес. Од друга страна, експлицитните податоци произлегуваат од можноста корисникот да ги рангира филмовите на скала од 0 до 10, што овозможува поконкретна оценка на неговите преференции.
Оваа двојност воведува комплексност во дефинирањето на евалуацијата што најреално би го прикажала успехот на моделот. Имено, корисникот не секогаш експлицитно оценува филмови што ги гледал, а истовремено постојат филмови што никогаш не ги гледал – не затоа што не му се интересни, туку затоа што не ги сретнал. Ова укажува на потребата од модел што ќе биде способен да ги разликува филмовите кои корисникот нема интерес да ги гледа од оние што би можел да ги гледа, но не бил изложен на нив.
Со цел да се постигне подобра персонализација, моделот треба да ги рангира филмовите така што оние најслични на веќе гледаните добиваат повисоки оценки. Овој проблем природно води кон pairwise loss пристап, каде што моделот учи да споредува парови филмови и да осигура дека поблиските до преференците на корисникот (во тренирањето, оние гледани од него) ќе бидат рангирани повисоко. Таков пристап е широко прифатен за системи со имплицитни податоци.
Во дел од моделите се користи pointwise loss - средна квадратна грешка, во кои се оптимизира предвидувањето на експлицитниот рејтинг, односно се проценува веројатноста дека даден филм ќе добие конкретна оценка од корисникот. Овој пристап овозможува поедноставно внесување на контекст за item-от во прашање, како опис на филмот, или жанр.

Со комбинирање на моделите тренирани со pairwise loss и pointwise loss, се овозможува комбинираниот модел да ги искористи предностите на двата пристапа, овозможувајќи подобро рангирање на филмовите, како и попрецизно предвидување на нивниот рејтинг.

5 Модели

Сите модели се имплементирани во python програмски јазик, користејќи библиотека tensorflow 2.17.0. Моделите се изградени со функционално keras API. [4]

5.1 Базичен NeuMF модел

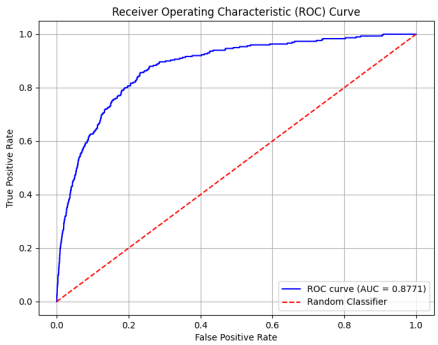


Слика 3: Незадоволителен резултат од добиениот модел после 10 епохи. Hit rate @10 : 0.446 AUC : 0.704

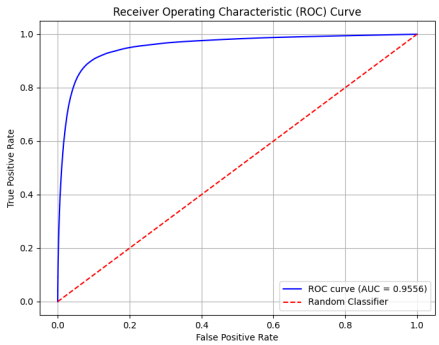
```
user_input = Input(shape=(1,), name='user_input')
item_input = Input(shape=(1,), name='item_input')

user_embedding = Embedding(n_users, embedding_dim, name='user_embedding')(user_input)
item_embedding = Embedding(n_items, embedding_dim, name='item_embedding')(item_input)

user_vec = Flatten()(user_embedding)
item_vec = Flatten()(item_embedding)

gmf_vec = Multiply()([user_vec,item_vec])
concat_vec = Concatenate()([user_vec, item_vec])
mlp = concat_vec
for size in mlp_layer_sizes:
    mlp = Dense(size, activation='relu')(mlp)
    mlp = Dropout(0.2)(mlp)

pre_output_concatenate = Concatenate()([gmf_vec,mlp])
output = Dense(1, activation='linear', name='output')(pre_output_concatenate)

ncf_model = Model(inputs=[user_input, item_input], outputs=output)
ncf_model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])

ncf_model.summary()
```

Може да се заклучи дека NeuMF моделот дава премногу лоши резултати на претходно дефинираната евалуација. Можни се две техники за да се добие соодветно негово функционирање:

1. **Претренирање на посебните GMF и MLP слоеви на истиот проблем и со истата loss функција**
2. **Замена на loss функцијата и работењето на моделот со таква што уште при тренирањето би внела во моделот подобро имплицитно рангирање на видените производи**



(а) Претториран GMF MLP модел со MSE loss



(б) GMF MLP модел со BPR loss

Слика 4: Споредба на двата најдобри модели

5.2 NeuMF модел so pointwise loss - Bayesian Personalized Ranking

Во реални услови, далеку поважната цел за квалитетното работење на моделот е релативното рангирање на производи кои корисникот е поверојатно да е заинтересиран за (а ги нема сретнато), отколку експлицитно рангирање на сите производи на некоја скала. За целосното множеството на производи во системот I може да се дефинира проблемот на рангирање како [3]:

$$
\begin{aligned}
&>_u \subset I \times I \\
\forall i,j \in I: \quad & i \neq j \rightarrow (i >_u j \vee j >_u i) \quad \text{(Тоталност)} \\
\forall i,j \in I: \quad & (i >_u j) \rightarrow \neg(j >_u i) \quad \text{(Асиметричност)} \\
\forall i,j,k \in I: \quad & (i >_u j \wedge j >_u k) \rightarrow (i >_u k) \quad \text{(Транзитивност)}
\end{aligned}
$$

Идејата е да се добие тоталното рангирање брз база на создавање множество од парови на позитивни и негативни производи придружени на соодветниот корисник (на позитивниот производ), при што е неопходно да се максимизира веројатноста моделот да го „избере“ производот со кој корисникот веќе имал интеракција. Авторите на [3] со таа цел воведуваат Bayesian Personalized Ranking, која во GMF-MLP е имплементирана како:

$$loss = -\frac{1}{N}\sum_{i=1}^{N} log(\sigma(s_i^+ - s_i^-)) \tag{2}$$

, каде s_i^+ и s_i^- претставуваат скорот на производот во и надвор од множеството производи со кои имал интеракција корисникот.
Важно е да се напомене дека моделот трениран на овој начин конвергира многу побрзо (во само 2 епохи на ова множество и со истите хиперпараметри како другите модели), особено земајќи в предвид дека не е потребно предтренирање за да се постигне повеќе од задоволителни резултати.

Модел	Базичен Модел	Претрениран	BPR loss модел
Hit Rate @ 10	0.446	0.6733	0.9920
NDCG @ 10	0.291	0.4019	0.9879
ROC AUC	0.701	0.8771	0.9556

Слика 5: Споредба на перформансите на моделите.

6 Збогатување на моделот со контекстни информации

Според резултати добиени од споредбата на моделот трениран со средно квадратна грешка како loss функција и Bayesian Personalized Ranking, може да се заклучи дека моделот специјализиран за погодување на веќе сретнати филмови од корисникот во општ случај претставува солиден избор. Моделот може да ги влоши перформансите во таканаречено „cold start“ сценарио, во кое ништо не е познато околу историјата на корисникот, па не би имал информации според кои би направил соодветно рангирање на информациите. За ова сценарио, предложено е да се преземат техники од моделите кои експлицитно рангираат филмови:

1. Генеричко рангирање на филмовите според просечниот ранкинг и популарност, подготвено за почетокот на искуството на корисникот со сајтот

2. Збогатување на информациите достапни на MSE моделите. Во конкретниот случај, испробано внесување на информации околу жанрот и популарноста на филмот директно во моделот (при конкатенацијата) → незначително подобрување на перформансите за значителна цена

3. Користење на елементи од безнадзорно (unsupervised) учење : претходно пресметани embedding вектори за филмовите (со кој било од претходните модели), тривијална имплементација на K Nearest Neighbors врз нив. Корисно во случај кога веќе се познати некои информации околу корисникот, но не доволно за ефективно рангирање. Испробано во проектот со векторизирање на overview текст за секој филм, истренирано на пример од 50000 од податочното множество. Незадоволителни резултати на валидација за премногу компутациски комплексен модел во услови на продукција/online работење на моделот.

Литература

[1] https://www.kaggle.com/datasets/samlearner/letterboxd-movie-ratings-data. Accessed: 2025-02-02.

[2] He et al. "Neural Collaborative Filtering". **in***arXiv*: (2017).

[3] Rendle et al. "BPR: Bayesian Personalized Ranking from Implicit Feedback". **in***UAI*: (2009), **pages** 452–461.

[4] Chollet. *The Functional API*. https://keras.io/guides/functional_api/. Accessed: 2025-02-02.