Recommender Systems

Gregor Laarmann

- Was sind Recommender Systems?
- Wieso werden diese benötigt?
- Welche Ansätze gibt es?
- Meine Herangehensweise?
  - Meine Daten (Netflix Prize Dataset)
- Mein Recommender
- Optimierungsmöglichkeiten
- Fazit und Github

Was ist ein Rec Sys?

- Empfehlungssystem
- Bestimmt, wie groß das Interesse eines Benutzers an einem Objekt ist
- Filtern relevante Informationen aus riesigen Datenmengen
- Bekämpfen das Problem der Informationsüberflutung
- Verwenden vorhandene Daten um Vorhersagen zu machen
    - Geschlecht, Alter, Region, Suchhistorie etc.
- Musik, Videos, Filme, Serien, Produkte uvm.

Warum Rec Sys?

- Interessante neue Produkte finden (E-Commerce)
- Gute User Experience
- Zeit sparen
- Hohe “Kundenbindung” erreichen

Mögliche Ansätze

- Collaborative Filtering
  - Empfehlungen basieren auf anderen Usern.
- Content Filtering
  - Empfehlungen werden getroffen anhand der Objekt-Eigenschaften.
- Matrix Factorization mit Dot Product (Skalarprodukt)
  - Matrix wird aufgeteilt in User & Movie-Matrix mit Latent Features, aus diesen Latent Features wird ein Skalarprodukt gebildet, welches dann der Prediction entspricht.
- Deep Hybrid System mit Movie-Metadata(TF-IDF)
  - Wörter in Filmbeschreibung werden gewichtet. Wörter, die häufig in einer Beschreibung vorkommen, in anderen aber nicht, werden höher gewichtet.
  - Dadurch können einzigartige Merkmale identifiziert werden, die die Relevanz des Films für einen bestimmten Benutzer bestimmt.

Meine Herangehensweise

- Tutorial als Template
  - Parametern gedreht, wie Epochen, Datensatzgröße, Größe von Test-und Trainingssatz
- Neuronale Netze mit Collaborative Filtering-Ansatz
  - Findet Muster in Daten, wie Benutzer Filme bewerten
- Tensorflow Keras API
- Root Mean Squared Error (RMSE)
  - Vergleicht die vorhergesagten Bewertungen mit den “echten” Bewertungen.
  - Je niedriger, desto höher die Genauigkeit.
- Netflix Prize-Dataset

Mein Datensatz (Netflix Prize-Dataset)

- Ca. 100 Millionen Bewertungen
- 480.000 Benutzern
- 17.770 Filme
- Datensatz ist geviertelt (4x ca. 500 MB)
- Gewinner 2009: 0.8567 RMSE

Shape User-Ratings: (22601629, 4)

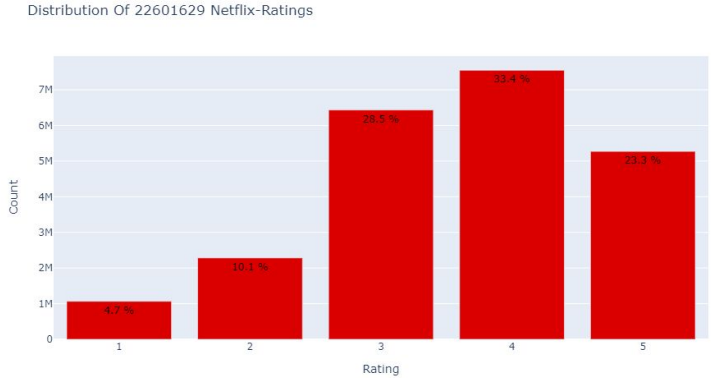
	User	Rating	Date	Movie
15461737	63486	4.0	2005-05-06	12155
6763782	2460211	2.0	2005-04-30	10583
12033635	748312	3.0	2003-05-29	11464
10498516	2067423	3.0	2004-11-24	11199
13718376	1573203	5.0	2005-09-03	11812

1. Alle Bewertungen

Shape Movie-Titles: (17770, 2)

ID	Year	Name
9806	1975.0	Bucktown
2019	2004.0	Samurai Champloo
6418	2000.0	Tigerland
2720	1992.0	Return to the Lost World
1784	1936.0	Camille

2. Alle Filme



3. Verteilung der Bewertungen

```
Shape User-Ratings unfiltered:  (22601629, 4)
Filtered movies with < 10k ratings and users with < 200 ratings
Shape User-Ratings filtered:    (3380856, 4)
```

4. Daten vor und nach Filter

Shape User-Movie-Matrix: (17474, 463)

Movie	9229	9232	9234	9235	9236	9240	9241	9242	9254	9265	...	13244	13251	13255	13273	13293	13298	13302	13318	13342	13359
User																					
2143870	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	NaN	3.0	...	2.0	NaN	4.0	2.0	NaN	NaN	4.0	NaN	NaN	4.0
2289273	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	...	NaN	NaN	NaN	4.0	NaN	NaN	4.0	NaN	NaN	4.0
1556884	NaN	NaN	NaN	3.0	NaN	NaN	NaN	4.0	4.0	NaN	...	NaN	3.0	4.0	NaN	NaN	NaN	4.0	4.0	NaN	5.0
2632429	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	4.0	3.0	...	3.0	NaN	NaN	NaN	4.0	NaN	2.0	3.0	4.0	NaN
1510334	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN

5 rows × 463 columns

5. User-Movie Matrix

Mein Recommender

- Input Layer
  - User_ID und Movie_ID
- Embedding Layer
  - Input Daten werden in verdichtete Vektoren umgewandelt
- Dense Layer
  - Nehmen die dichten Vektoren als Input
  - Lernt Patterns zwischen User und Movie Embeddings, um Vorhersagen zu treffen
- Output Layer
  - Predictions als Output
  - Top 10 Output zeigen

Top 10 movies

```
Top 10 recommended movies for user 431116:
Lord of the Rings: The Return of the King: Extended Edition
The Lord of the Rings: The Fellowship of the Ring: Extended Edition
Lord of the Rings: The Two Towers: Extended Edition
The Shawshank Redemption: Special Edition
Star Wars: Episode V: The Empire Strikes Back
The Simpsons: Season 5
The Simpsons: Season 4
Lord of the Rings: The Return of the King
Band of Brothers
The Sopranos: Season 1
```

Ansätze/RMSE	3.3 Mio Datensätze 5 Epochs Testset: 100k/ 1 Mio/ 660k	24.1 Mio Datensätze 5 Epochs Testset: 100k/1 Mio/ 4.8 Mio	3.3 Mio Datensätze 20 Epochs Testset: 100k/1 Mio/660k	24.1 Mio Datensätze 20 Epochs Testset: 100k/1 Mio/4.8 Mio
Matrix Factorization	0.8639 RMSE (221s) 0.8739 RMSE (194s)	 0.8301 RMSE (1378s)		
Keras mit DL	0.8959 RMSE (179s) 0.9014 RMSE (210s) 0.9019 RMSE (213s)	0.8887 RMSE (2090s) 0.8893 RMSE (1821s) 0.8890 RMSE (1730s)	0.9021 RMSE (ca. 840s) 0.9007 RMSE (ca. 650s) 0.8997 RMSE (ca. 740s)	 0.8878 RMSE (ca. 30000s) ohne GPU 0.8878 RMSE (ca. 5500s)
Hybrid	0.9915 RMSE (932s)			

Vergleich der Ergebnisse

Optimierungsmöglichkeiten

- Trial und Error Ansatz beim Modifizieren der Daten
- Mehr Kriterien für die Predictions berücksichtigen
- Mehr Daten = genauere Ergebnisse

Fazit und Github

- Zeitplan nicht wirklich eingehalten
  - Keine Erfahrung mit ähnlichen Projekten
  - Trotzdem versucht noch weitere Ansätze kurz anzusprechen und zu vergleichen
- Erstes Machine Learning Projekt
- Man kann unendlich Tief in die Materie gehen
  - Abschlussarbeit
- Im Großen und Ganzen hat es spaß gemacht
- Github: https://github.com/noobgreg/DL_Rec_Sys