Multifaceted Collaborative Filtering Model

SMAI CSE 471
Spring 2019
Dr. Ravi Kiran Sarvadevabhatla

Team Name: Team Houdini (33)

Mentor: Nikhil Gogate

Outline

- Problem Statement
- What are Recommender Systems ?
- Motivation
- Dataset
- Models
  - Neighborhood Model
  - SVD++ Model
  - Integrated Model
- Results
- Challenges and Limitations

Problem Statement

- To use collaborative filtering techniques to apply on Movie dataset that recommends the movies for users based on the reviews and past data.
- Implement baseline CF models
  - Neighborhood model
  - SVD++ model
- Improve them using technique that integrates the two models.

What are Recommender Systems ?

- Information filtering system that seeks to predict the "rating" or "preference" a user would give to an item.
- Two main types of Recommender Systems:
  - Content based
  - Collaborative Filtering
    - Neighborhood Model
    - SVD++(Latent Factor Model)
- Used in variety of areas:
  - Video and music recommenders (Netflix, YouTube, Spotify)
  - Product recommenders (Amazon, Myntra)

Motivation

- Enhancing user satisfaction and loyalty by matching consumers with appropriate products.
- Netflix Prize - Open competition for best CF algorithm to predict user rating for films.
- “We need to go win a million dollars”

Dataset

- MovieLens 100k data.
- Collected by the GroupLens Research Project at the University of Minnesota.
- 100,000 ratings (1-5) from 943 users on 1682 movies.
- Each user has rated at least 20 movies.

```
u_data.head()
```

	user_id	movie_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

Fig1 : Dataset entries

Dataset - EDA

Fig2 : Rating Distribution

Dataset - EDA

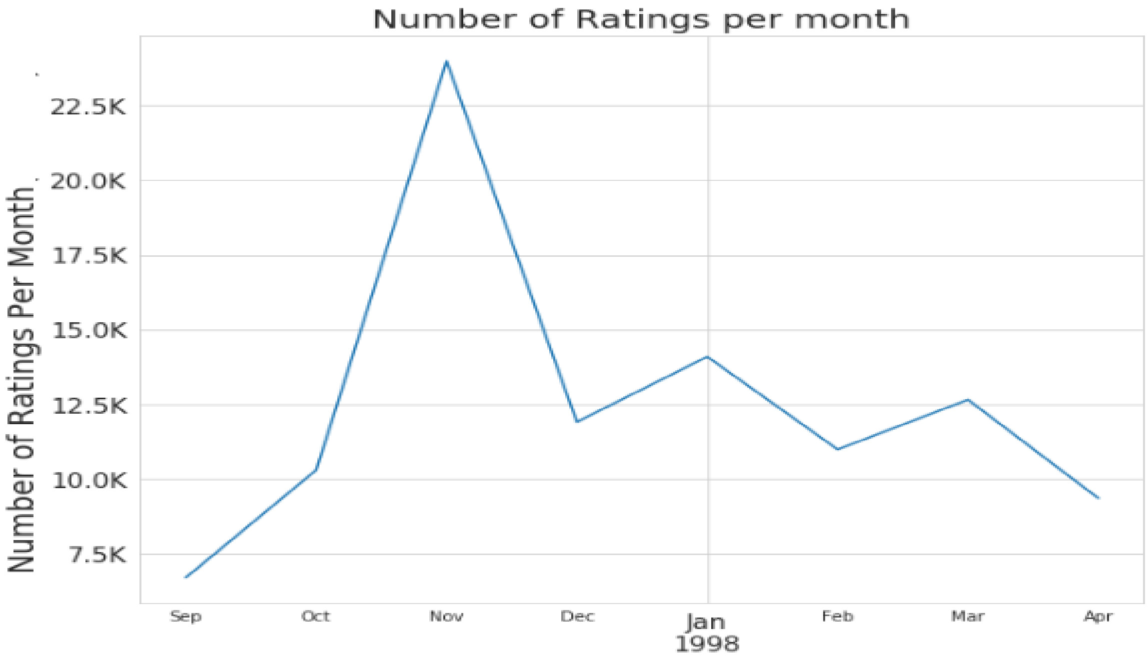


Fig3 : Rating Distribution

Dataset - EDA

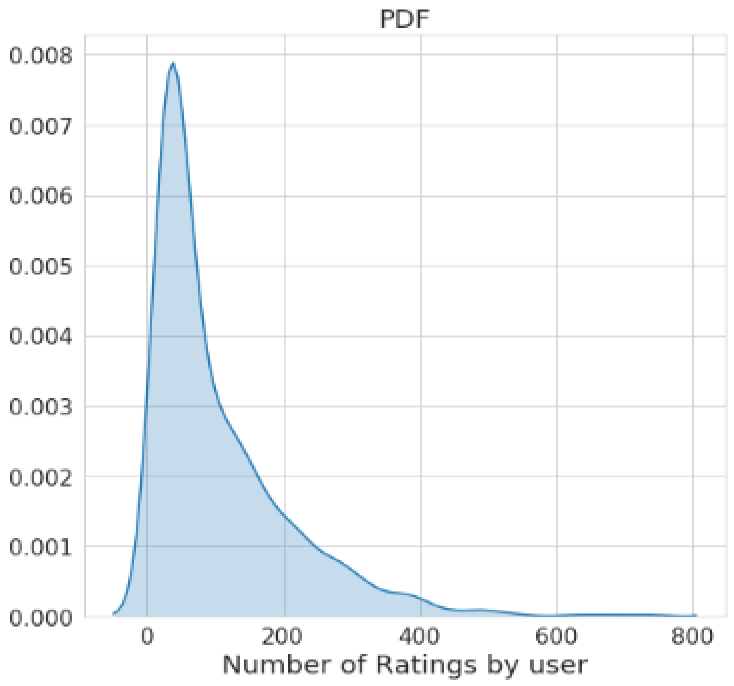


Fig4 : PDF of Number of Ratings by user

- PDF graph shows that almost all of the users give very few ratings. There are very few users who's ratings count is high.

Dataset - EDA

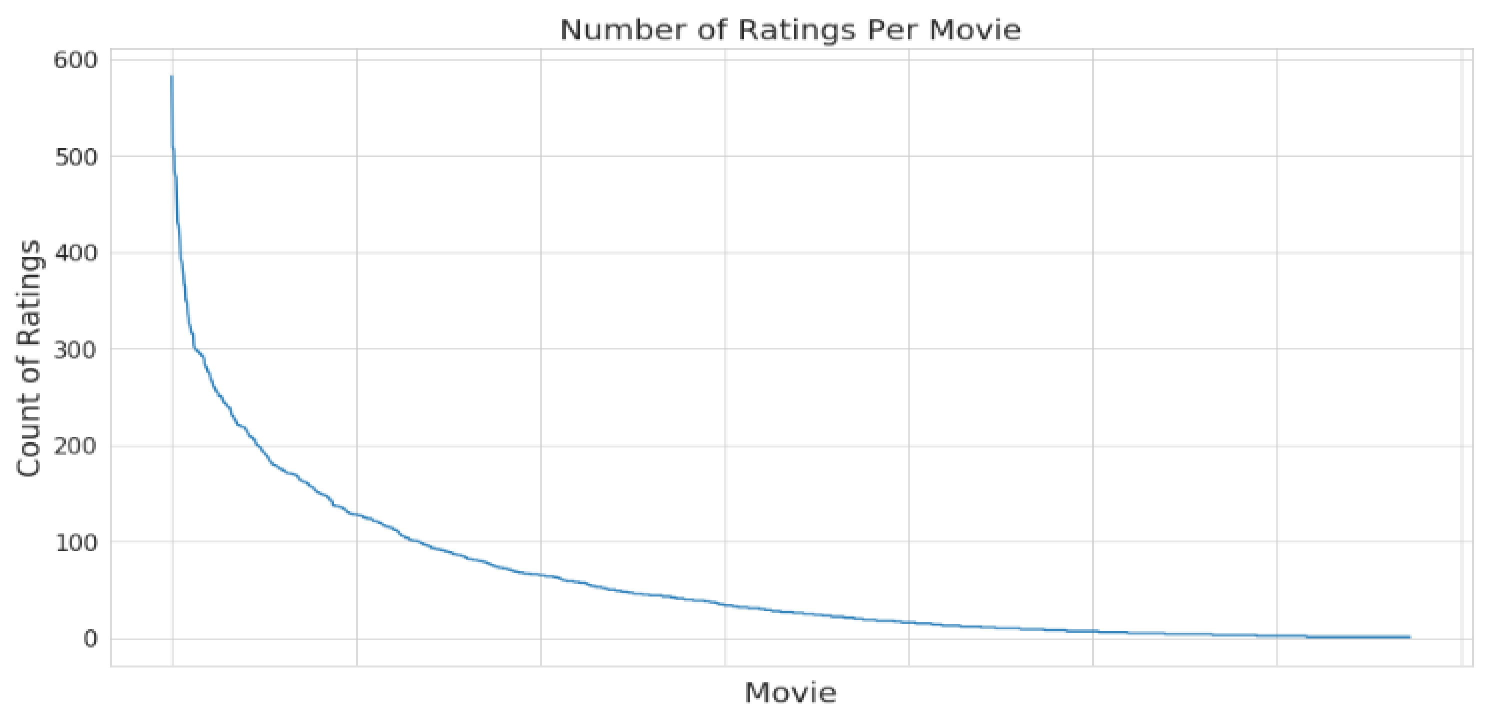


Fig5 : Number of Ratings per Movie

- Some movies are very popular and rated by many users vs other movies.

Dataset - EDA

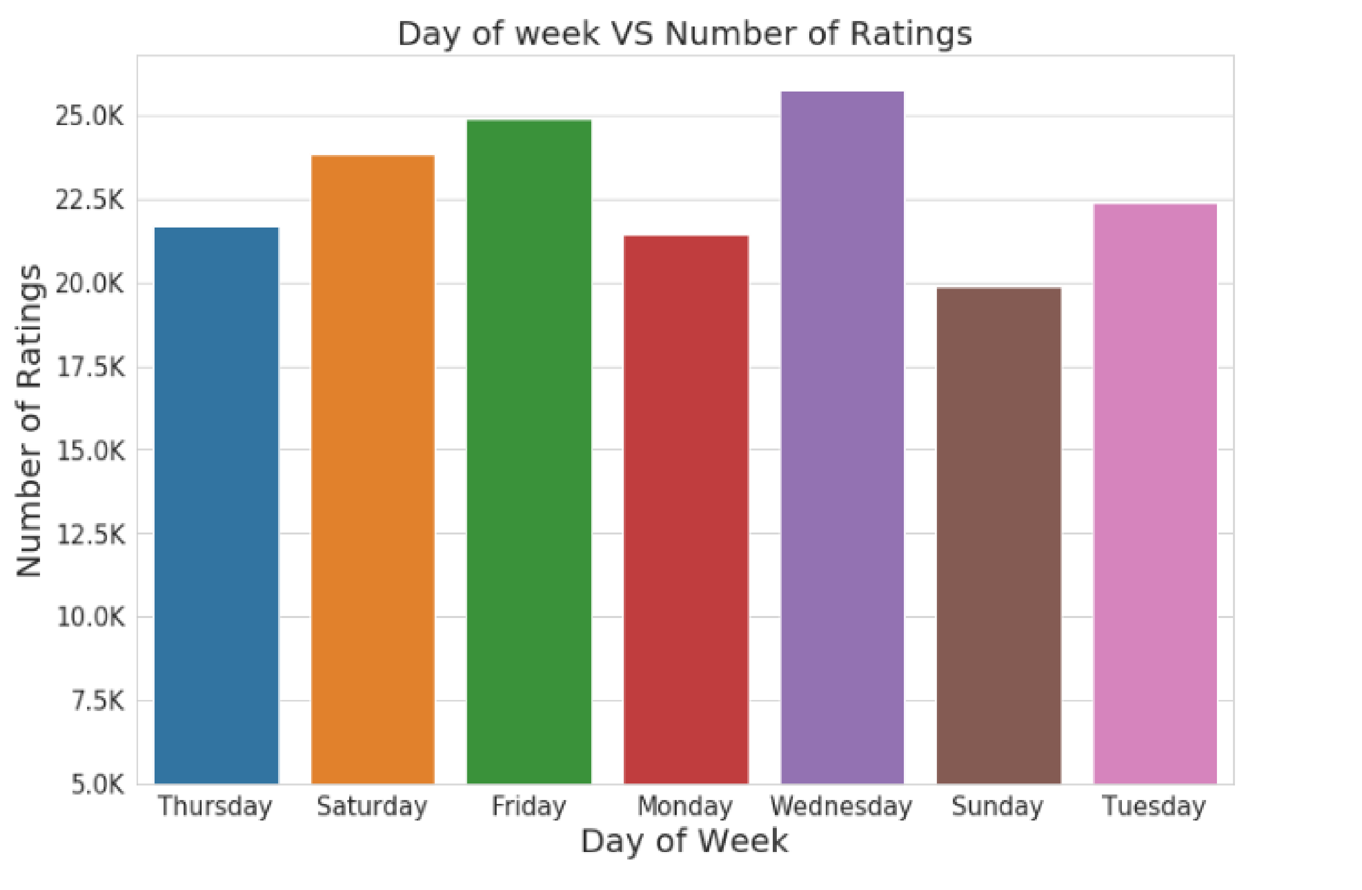


Fig6 : Day of Week v/s Number of Ratings

Baseline estimate

- Baseline estimate for predicting rating for movie i by user u (b_{ui})

$$b_{ui} = \mu + b_u + b_i$$

  - Item bias (b_i)
  - Rating by user u for item i (r_{ui}).

- Implicit feedback (N(u)) contains all items for which implicit preference was provided by user u.)
- In order to estimate bu and bi we can solve the least squares problem :

$$\min_{b_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 \left(\sum_u b_u^2 + \sum_i b_i^2\right)$$

SVD and SVD++ model

- **Matrix factorization** is a class of collaborative filtering algorithms.
- A popular approach to latent factor models is induced by an SVD-like lower rank decomposition of the ratings matrix.
- Each user u is associated with a user-factors vector $p_u \in R_f$, and each item i with an item-factors vector $q_i \in R_f$.
- Prediction is done by the rule: $\hat{r}_{ui} = b_{ui} + p_u^T q_i$
- This is the SVD model. An improvement to this model is Asymmetric SVD which uses implicit feedback.
- As we do not really have much independent implicit feedback for the our ml-100k dataset, so we turn towards an improved model.

SVD and SVD++ model

SVD++ model :

$$\hat{r}_{ui} = b_{ui} + q_i^T \left(p_u + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j \right)$$

- Its results are more accurate than all previously published methods on the Netflix data and other similar movie datasets which struggles with the same implicit feedback limitation.

Neighborhood model

- User oriented CF system.
- Estimate unknown ratings based on recorded ratings of like-minded users.
- Neighborhood model :

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in \mathrm{S}^k(i;u)} \theta_{ij}^u (r_{uj} - b_{uj})$$

- Improved Neighborhood model as described by the equation :

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) w_{ij}$$

Neighborhood model

- We can use implicit feedback, which provide an alternative way to learn user preferences. To this end, we add another set of weights, and rewrite the previous equation :

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) w_{ij} + \sum_{j \in \mathrm{N}(u)} c_{ij}$$

- Final Model :

$$\hat{r}_{ui} = \mu + b_u + b_i + |\mathrm{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) w_{ij} + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} c_{ij}$$

Integrated model

- A combined model which will sum the predictions of previously defined neighborhood and SVD++ model.

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j \right)$$
$$+ |\mathrm{R}^k(i;u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + |\mathrm{N}^k(i;u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}^k(i;u)} c_{ij}$$

Integrated model

- Backprop for Integrated model :

  - $b_u \leftarrow b_u + \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_u)$
  - $b_i \leftarrow b_i + \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_i)$
  - $q_i \leftarrow q_i + \gamma_2 \cdot (e_{ui} \cdot (p_u + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j) - \lambda_7 \cdot q_i)$
  - $p_u \leftarrow p_u + \gamma_2 \cdot (e_{ui} \cdot q_i - \lambda_7 \cdot p_u)$
  - $\forall j \in \mathrm{N}(u)$:
    $y_j \leftarrow y_j + \gamma_2 \cdot (e_{ui} \cdot |\mathrm{N}(u)|^{-\frac{1}{2}} \cdot q_i - \lambda_7 \cdot y_j)$
  - $\forall j \in \mathrm{R}^k(i; u)$:
    $w_{ij} \leftarrow w_{ij} + \gamma_3 \cdot \left(|\mathrm{R}^k(i; u)|^{-\frac{1}{2}} \cdot e_{ui} \cdot (r_{uj} - b_{uj}) - \lambda_8 \cdot w_{ij} \right)$
  - $\forall j \in \mathrm{N}^k(i; u)$:
    $c_{ij} \leftarrow c_{ij} + \gamma_3 \cdot \left(|\mathrm{N}^k(i; u)|^{-\frac{1}{2}} \cdot e_{ui} - \lambda_8 \cdot c_{ij} \right)$

Implementation Insights

- We've implemented all three models mentioned
  - Neighborhood Model
  - SVD++ Model
  - Integrated Model
- Datasets was around 99% sparse, so we used Sparse Matrix (CSR format) instead of Dense Matrix.
- One assumption made was that every user who have watched the movie has rated it.
- Number of Latent factors for user and item used were 2**0** and Epoch count was **30**

Implementation Insights

- Parameters used :
- Meta parameters: $\gamma 1 = \gamma 2 = 0.007$, $\gamma 3 = 0.001$, $\lambda 6 = 0.005$, $\lambda 7 = \lambda 8 = 0.015$.
  - We decrease step sizes (the γ's) by a factor of 0.9 after each iteration.
  - All results are measured for Epochs of 30.
  - Epoch time for running the models on Kaggle/Laptop(Mac) were :
    - Neighborhood Model : 2.30 mins (Mac)
    - SVD++ model : 2.13 mins (Kaggle)
    - Integrated model : 3.03 mins (Kaggle)

Results

```
 processing epoch 28
Time For Epoch :: 0:02:41.989238
Err =  1.7757690045029083
Time For Error :: 0:00:08.660341
 processing epoch 29
Time For Epoch :: 0:02:24.828145
Err =  1.775727153704525
Time For Error :: 0:00:08.595317
1.775727153704525
```

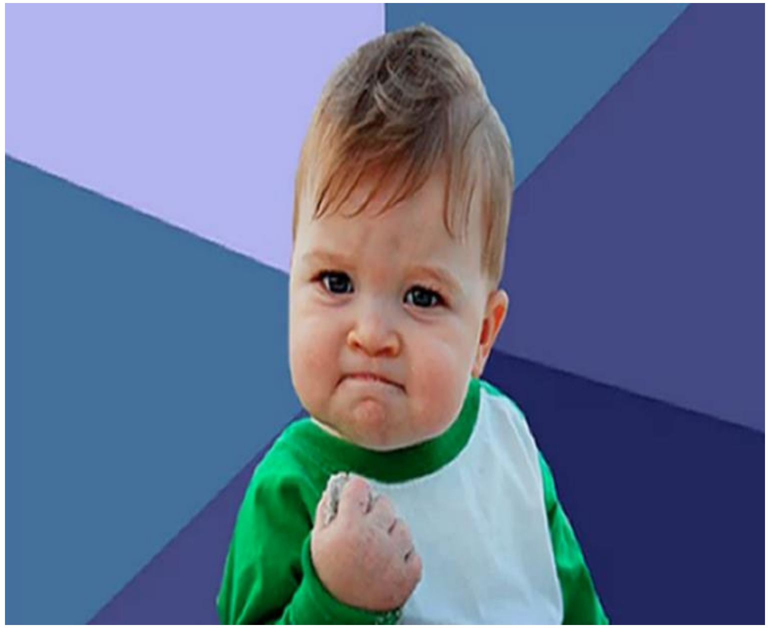
RMSE Error (Neighborhood Model)

```
 processing epoch 28
Time For Epoch :: 0:02:13.639239
Err =  0.9411261391466791
Time For Error :: 0:00:04.046057
 processing epoch 29
Time For Epoch :: 0:02:13.121249
Err =  0.941076217885924
Time For Error :: 0:00:04.004687
0.941076217885924
```

RMSE Error (SVD++ Model)

Results

```
 processing epoch 28
Time For Epoch :: 0:03:02.497476
Err =  0.928370015679691 8
Time For Error :: 0:00:06.288783
 processing epoch 29
Time For Epoch :: 0:03:02.442041
Err =  0.928305834760709
Time For Error :: 0:00:06.636619
0.928305834760709
```



RMSE Error (Integrated Model)

Neighborhood Model	SVD++	Integrated Model
1.7757	0.941	0.9283

Error in RMSE (root mean square error)

Predictions

```
print("Watched movies by user: "+usr)
print()
for i in seen_movie_str_list:
    print(i)
```

```
Watched movies by user: 203

Hercules (1997)::  Adventure   Animation  Childrens   Comedy   Musical
Starship Troopers (1997):: Action   Adventure   Sci-Fi   War
One Fine Day (1996)::  Drama   Romance
Nixon (1995)::  Drama
Mother (1996)::  Comedy
Star Trek: First Contact (1996):: Action   Adventure   Sci-Fi
Emma (1996)::  Drama   Romance
Ransom (1996)::  Drama  Thriller
Fly Away Home (1996)::  Adventure  Childrens
Playing God (1997)::  Crime  Thriller
```

```
Predicted movies for user: 203
Fargo (1996)::  Crime   Drama  Thriller
Return of the Jedi (1983):: Action   Adventure   Romance   Sci-Fi   War
Michael Collins (1996)::  Drama   War
Willy Wonka and the Chocolate Factory (1971)::  Adventure  Childrens   Comedy
Scream (1996)::  Horror  Thriller
Saint::  Adventure   Sci-Fi   War
Liar Liar (1997)::  Comedy
```

Limitations

- Insufficient hardware support to run large dataset (Netflix dataset), even in CSR format.
- Better sources needed for implicit feedback.
- Data sparsity
- Scalability
- Cold Start is genuine problem for Recommender Models. It's relevant for both new users and new movies which the model encounters.
- For our model :
  - If the { User , Movie } pair is new to the model, we predict the global mean of all the movies.

Thank You.