Movie Recommendation System

Daniel Ying dty16 (Group 27)

Table of Content

- Implement, Experiment, and Compare different models.
- Design, Implement a Hybrid Model (NCF)
  - Evaluate the performance
- Conclusion

About the Data

- The data (ml-latest-small) used in the project is from GroupLens, a research group in Department of Computer Science and Engineering at the University of Minnesota.
- The 4 csv files:
  - Ratings.csv (Main Data file), movies.csv, tags.csv, links.csv
- Ratings.csv has the following data:
  - UserId
  - MovieId
  - Rating
  - Timestamp

More about the data…

```
ratings shape: (100836, 4)
first 5 ratings:
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

- The Ratings.csv has the shape of (100836, 4)
- 610 unique users, and 9724 unique movies
- For this project, 80% of the data is randomly selected to be the training set, while the other 20% of the data is defined as the testing set for the models.

Models Used

- Models From the Surprise Library:
  - SVD, NMF, SlopeOne, CoClustering, BaselineOnly
  - KNN Series
- The Core Model: Neural Collaborative Filtering Model
  - Generalized Matrix Factorization + Multi-Layer Perceptron

Surprise Models

- SVD: (Singular Value Decomposition)used as CF in recommendation sys. It reduces the number of features of the dataset by reducing space dimension:
  - $A = P\sum Q^T$
  - $\hat{r_{ui}} = \mu + b_i + b_u + (q_i)^T p_u$, the predicted rating r and mu is average rating while b is deviation, q is for movie concept while p is for user concept.
- NMF: Non-negative Matrix Factorization
  - $\hat{r_{ui}} = q_i^T p_u$
- Baseline: predicts baseline estimation of user and movie:
  - $\hat{r_{ui}} = \mu + b_i + b_u$

Surprise Models

- SlopeOne: simple CF that is based on the linear relation between the user and the movie. Uses the linear regression model to predict the rating.
  - $\hat{r_{ui}} = \mu_u + \frac{1}{|R_i(u)|} \sum_{\{j \in R_i(u)\}} dev(i,j)$
  - Riu is set of relevant items (movies) and j is rated by u (user) that also have at least one comment user i.
  - dev is the average difference between ratings by user I and user j.

Surprise Models

- CoClustering: users are assigned clusters Cu, Ci, and co-clusters Cui.
  - $\hat{r_{ui}} = avg(C_{ui}) + \left(\mu_u - avg(C_u)\right) + (\mu_i - avg(C_i))$
  - Avg(Cui) is average rating of Cui, and same for the rest clusters.
  - If user is unknown, rui = mu_i.
  - If item is unknown, rui = mu_u.
  - If both are unknown, rui = mu.

Surprise KNN Models

- Basic KNN: takes account of max number of k neighbors and similarity metrics as input
- KNN with Means: takes into account the mean ratings of each users into account
- KNN with Z-scores: takes in account of z-normalization of each users
- KNN baseline: takes in account the baseline rating
- KNN metrics:
  - Cosine: measure similarity between 2 non-zero vectors
  - Mean square difference: inversely proportional to difference of users' mean square difference.
  - Pearson: mean-centered cosine similarity
  - Pearson Baseline: baseline-centered cosine similarity
- User-user and Item-item similarities in KNN.

Results of Models

Measures

- RMSE: root mean square error
- MAE: mean absolute error
- Precision: (relevant recommended items)/(total rec. items)
- Recall: (relevant rec. items)/ (total relevant items)
- F-Measure: 2(precision* recall) / (precision+ recall), best is 1
- NDCG: normalized discounted cumulative gain (wiki), sort all relevant documents by relative relevance, produce max possible DCG (aka ideal for that position p)

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2(i+1)}$$

$$\mathrm{IDCG_p} = \sum_{i=1}^{|REL_p|} \frac{rel_i}{\log_2(i+1)}$$

$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p},$$

Results From Surprise Models

KNNBaseline Item Pearline	0.6544214791312891
KNNBaseline Item MSD	0.6686846948739565
KNNBaseline User Pearline	0.6726476067823245
KNNBaseline User MSD	0.6728208025621423
SVD	0.6748513010089627
KNNBaseline User Pear	0.6758026468389403
KNNBaseline User Cos	0.6761500838166293
BaselineOnly	0.6763669135845803
KNNBaseline Item Pear	0.6791822334366855
KNNWithMeans User	0.6886208888687823
KNNWith Means Item	0.6886208888687823
KNNZ User	0.6886208888687823
KNNZ Item	0.6886208888687823
SlopeOne	0.6927328710255556
KNNBaseline Item Cos	0.6960748100521774
KNNBasic Item MSD	0.7028014662901718
NMF	0.7098917623418507
KNNBasic User MSD	0.7305797074310052
CoClustering	0.7350673342266314
KNNBasic User Cos	0.7526526548517168
KNNBasic Item Cos	0.7742228892538924

KNNBaseline Item Pearline	0.858280224151538
KNNBaseline Item MSD	0.872836190393700
BaselineOnly	0.876689030302501
SVD	0.879544889916836
KNNBaseline User MSD	0.88158399508237
KNNBaseline Item Pear	0.883541800486748
KNNBaseline User Pearline	0.884051986277553
KNNBaseline User Cos	0.884943109330249
KNNBaseline User Pear	0.885017054928852
KNNBaseline Item Cos	0.903377924247580
KNNWithMeans User	0.903482744129088
KNNWith Means Item	0.903482744129088
KNNZ User	0.903482744129088
KNNZ Item	0.903482744129088
SlopeOne	0.906643008429340
KNNBasic Item MSD	0.914964601352633
NMF	0.929199477676719
CoClustering	0.950168887214654
KNNBasic User MSD	0.955627428794643
KNNBasic User Cos	0.979848184079456
KNNBasic Item Cos	0.995619956790904

	uid	iid	r_ui	est	details
0	1	157	5.0	3.917850	{'was_impossible': False}
1	1	216	5.0	4.191880	{'was_impossible': False}
2	1	296	3.0	4.870920	{'was_impossible': False}
3	1	333	5.0	4.502708	{'was_impossible': False}
4	1	349	4.0	4.308776	{'was_impossible': False}

- The Left data is MAE
- The right data is RSME
- Item-item has a lower value than user-user.

4 Measures of Surprise Models

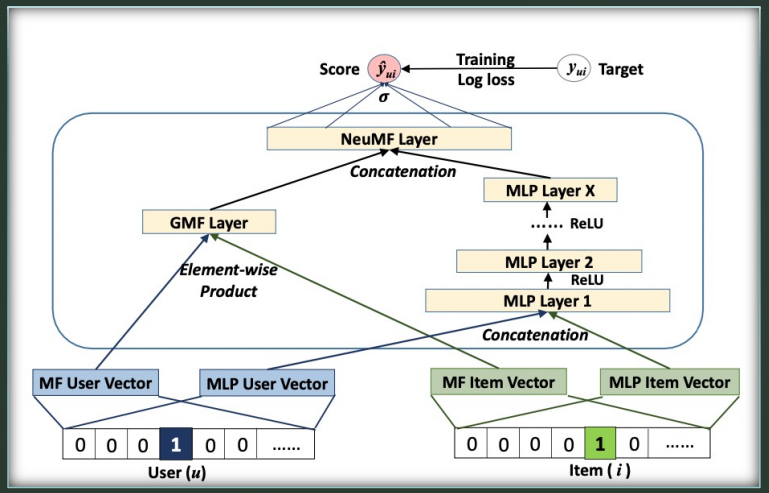
	Algorithm	Recall	Precision	F-Measure	NDCG
0	svd	0.631081	0.746740	0.684056	0.950120
1	slopeone	0.632387	0.747549	0.685163	0.946630
2	nmf	0.604784	0.718138	0.656605	0.943497
3	knnz-user	0.637244	0.749144	0.688678	0.941094
4	knnz-item	0.637244	0.749144	0.688678	0.941094
5	knnmean-user	0.637244	0.749144	0.688678	0.941094
6	knnmean-item	0.637244	0.749144	0.688678	0.941094
7	knnbasic-user	0.662519	0.797699	0.723852	0.948319
8	knnbasic-item	0.622308	0.730851	0.672226	0.941343
9	knnbaseline-user	0.645371	0.765710	0.700409	0.945723
10	knnbaseline-item	0.634790	0.750635	0.687869	0.951757
11	Coclustering	0.604260	0.716617	0.655660	0.943728
12	baseline	0.631700	0.751230	0.686299	0.951494

- Looking at KNN baseline-user and KNN baseline-item:
  - Item-item CF > user-user CF
- This might be because users have a tendency of changing their preference in movie selection.
- Thus movie-movie similarity is more consistent than user-user similarity.

NCF

What is NCF?

- NCF (Neural Collaborative Filtering):
  - Based on neural networks
  - Composed of GMF (Generalized Matrix Factorization) + MLP (Multi-Layer Perceptron)
  - The GMF and MLP learned user and movie embedding separately (allows tuning separately)



GMF

- GMF is defined as:
  - $dot(p_u, q_i)$, which is the element-wise product of vectors.
  - This is then projected as:
    - $a_{out}(h^T(dot(p_u, q_i))$
  - The term Generalized comes from:
    - The a_out which is the sigmoid function and learns h from the data with log loss.
  - Thus, GMF is defined as the dot product between the UserId and MovieId in the code.

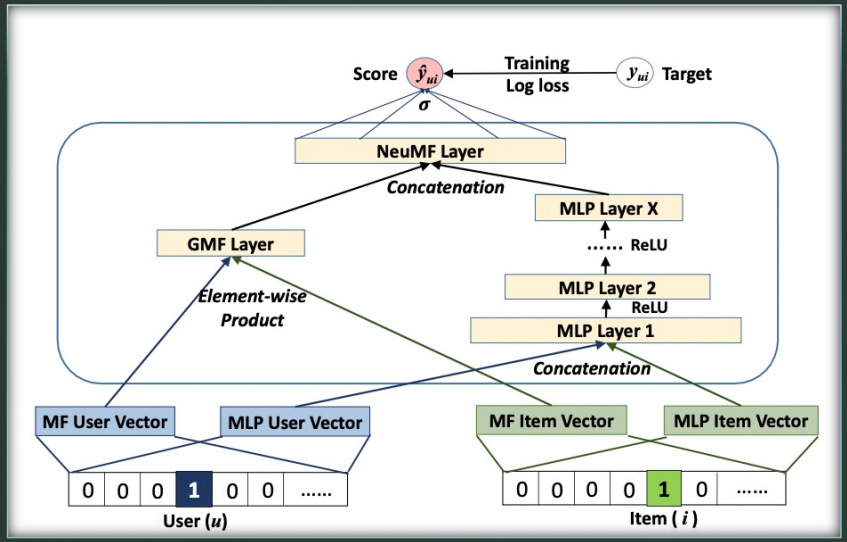
MLP

$$
\begin{aligned}
\mathbf{z}_1 &= \phi_1(\mathbf{p}_u, \mathbf{q}_i) = \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix}, \\
\phi_2(\mathbf{z}_1) &= a_2(\mathbf{W}_2^T \mathbf{z}_1 + \mathbf{b}_2), \\
&\cdots\cdots \\
\phi_L(\mathbf{z}_{L-1}) &= a_L(\mathbf{W}_L^T \mathbf{z}_{L-1} + \mathbf{b}_L), \\
\hat{y}_{ui} &= \sigma(\mathbf{h}^T \phi_L(\mathbf{z}_{L-1})),
\end{aligned}
$$

- Learn the relation between user and movie latent features
- hidden layers are added to the MLP to allow high level of flexibility and non-linearity.
  - The alpha: x-th layer activation function
  - W is the weight matrix,
  - b is the bias vector.
- The activation function can be sigmoid, tanh, and ReLU.
  - ReLU was implemented, for it encourages sparse activations (well-suited for sparse data)
  - thus prevent overfitting (sigmoid and tanh's flaw).
- Learning rate = 0.01

Combine GMF and MLP = NCF

- Concatenate GMF and MLP to obtain prediction of NCF



$$\phi^{GMF} = \mathbf{p}_u^G \odot \mathbf{q}_i^G,$$

$$\phi^{MLP} = a_L(\mathbf{W}_L^T(a_{L-1}(...a_2(\mathbf{W}_2^T \begin{bmatrix} \mathbf{p}_u^M \\ \mathbf{q}_i^M \end{bmatrix} + \mathbf{b}_2)...)) + \mathbf{b}_L),$$

$$\hat{y}_{ui} = \sigma(\mathbf{h}^T \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}),$$

(12)

NCF Results

NCF 4 measures

	Algorithm	Recall	Precision	F-Measure	NDCG
0	Neural Collaborative Filtering	0.7325553467000840	0.8661432226399330	0.7937679571638840	0.9389520705161180
1	Generalized Matrix Factorization	0.5960095551378440	0.6961244256474520	0.6421885275373240	0.9475637658465470

NCF_MAE_RSME_table

	Algorithm	MAE	RSME
0	NCF	0.679384940734982	0.9022068009032480
1	GMF	0.7751338284519320	1.0949647014682100

Comparison and Conclusion

NCF vs Models

NCF_MAE_RSME_table

	Algorithm	MAE	RSME
0	NCF	0.679384940734982	0.9022068009032480
1	GMF	0.7751338284519320	1.0949647014682100

KNNBaseline Item Pearline	0.6544214791312891
KNNBaseline Item MSD	0.6686846948739565
KNNBaseline User Pearline	0.6726476067823245
KNNBaseline User MSD	0.6728208025621423
SVD	0.6748513010089627
KNNBaseline User Pear	0.6758026468389403
KNNBaseline User Cos	0.6761500838166293
BaselineOnly	0.6763669135845803
KNNBaseline Item Pear	0.6791822334366855
KNNWithMeans User	0.6886208888687823
KNNWith Means Item	0.6886208888687823
KNNZ User	0.6886208888687823
KNNZ Item	0.6886208888687823
SlopeOne	0.6927328710255556
KNNBaseline Item Cos	0.6960748100521774
KNNBasic Item MSD	0.7028014662901718
NMF	0.7098917623418507
KNNBasic User MSD	0.7305797074310052
CoClustering	0.7350673342266314
KNNBasic User Cos	0.7526526548517168
KNNBasic Item Cos	0.7742228892538924

KNNBaseline Item Pearline	0.858280224151538
KNNBaseline Item MSD	0.872836190393700
BaselineOnly	0.876689030302501
SVD	0.879544889916836
KNNBaseline User MSD	0.88158399508237
KNNBaseline Item Pear	0.883541800486748
KNNBaseline User Pearline	0.884051986277553
KNNBaseline User Cos	0.884943109330249
KNNBaseline User Pear	0.885017054928852
KNNBaseline Item Cos	0.903377924247580
KNNWithMeans User	0.903482744129088
KNNWith Means Item	0.903482744129088
KNNZ User	0.903482744129088
KNNZ Item	0.903482744129088
SlopeOne	0.906643008429340
KNNBasic Item MSD	0.914964601352633
NMF	0.929199477676719
CoClustering	0.950168887214654
KNNBasic User MSD	0.955627428794643
KNNBasic User Cos	0.979848184079456
KNNBasic Item Cos	0.995619956790904

NCF vs Models

- NCF has better measures than the models.
- Outperforms the individual algorithms

	Algorithm	Recall	Precision	F-Measure	NDCG
0	svd	0.631081	0.746740	0.684056	0.950120
1	slopeone	0.632387	0.747549	0.685163	0.946630
2	nmf	0.604784	0.718138	0.656605	0.943497
3	knnz-user	0.637244	0.749144	0.688678	0.941094
4	knnz-item	0.637244	0.749144	0.688678	0.941094
5	knnmean-user	0.637244	0.749144	0.688678	0.941094
6	knnmean-item	0.637244	0.749144	0.688678	0.941094
7	knnbasic-user	0.662519	0.797699	0.723852	0.948319
8	knnbasic-item	0.622308	0.730851	0.672226	0.941343
9	knnbaseline-user	0.645371	0.765710	0.700409	0.945723
10	knnbaseline-item	0.634790	0.750635	0.687869	0.951757
11	Coclustering	0.604260	0.716617	0.655660	0.943728
12	baseline	0.631700	0.751230	0.686299	0.951494

NCF 4 measures

	Algorithm	Recall	Precision	F-Measure	NDCG
0	Neural Collaborative Filtering	0.7325553467000840	0.8661432226399330	0.7937679571638840	0.9389520705161180
1	Generalized Matrix Factorization	0.5960095551378440	0.6961244256474520	0.6421885275373240	0.9475637658465470

What's Next?

- Include other parameters? Genres?
- Other Hybrid Algorithms?
- A larger dataset? 1M data set? Complete Latest Dataset?

Reference

- F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19.
- Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, Tat-Seng Chua. 2017. Neural Collaborative Filtering. arXiv: 1708.0503lv2 [cs.IR]
- https://en.wikipedia.org/wiki/Discounted_cumulative_gain
- https://surprise.readthedocs.io/en/stable/index.html

Thank You!