Recommender system using matrix factorisation

Collaborative filtering on the MovieLens dataset

Patrick Indri
December 10, 2019

DSSC - Information Retrieval Exam

Contents

Introduction

Problem/assignment:

Build a recommender system that returns a ranked list of documents for each queried user.

Introduction

Problem/assignment:

Build a recommender system that returns a ranked list of documents for each queried user.

Adopted strategy:

Collaborative filtering using matrix factorisation. Only user behaviour will be used to make recommendations.

MovieLens dataset

The dataset (`ml-latest-small`) describes the 5-star rating activity from the MovieLens website, a movie recommendation service.

Dataset info:

- 9719 movies, rated by 610 users;
- Ratings from March 1996 to September 2018;
- Each user has rated at least 20 movies;
- For each rating, a timestamp is provided.

Other (larger) versions of the dataset are available for research and education purposes.

MovieLens dataset

The dataset (`ml-latest-small`) describes the 5-star rating activity from the MovieLens website, a movie recommendation service.

Dataset info:

- 9719 movies, rated by 610 users;
- Ratings from March 1996 to September 2018;
- Each user has rated at least 20 movies;
- For each rating, a timestamp is provided.

Other (larger) versions of the dataset are available for research and education purposes.

Why collaborative and not item-based filtering?

No need to know the number of genres, actors or other any other category.

Implicit vs explicit ratings

The selected dataset consists of an *explicit* feedback dataset.

Explicit vs implicit ratings:

- Explicit ratings are numerical scores (number of stars, rating on a numerical scale);
- Implicit ratings are non-obvious preferences, such as number of clicks, views, purchases, etc.

Real cases are usually implicit feedback ones.

Reading the dataset

Ratings and movies are stored in separate files. Unfortunately movie IDs do not increase continuously (IDs go up to 19K) and should be manually fixed. This avoids a massive amount of zeros.

Reading the dataset

Ratings and movies are stored in separate files. Unfortunately movie IDs do not increase continuously (IDs go up to 19K) and should be manually fixed. This avoids a massive amount of zeros.

Resulting dataframe:

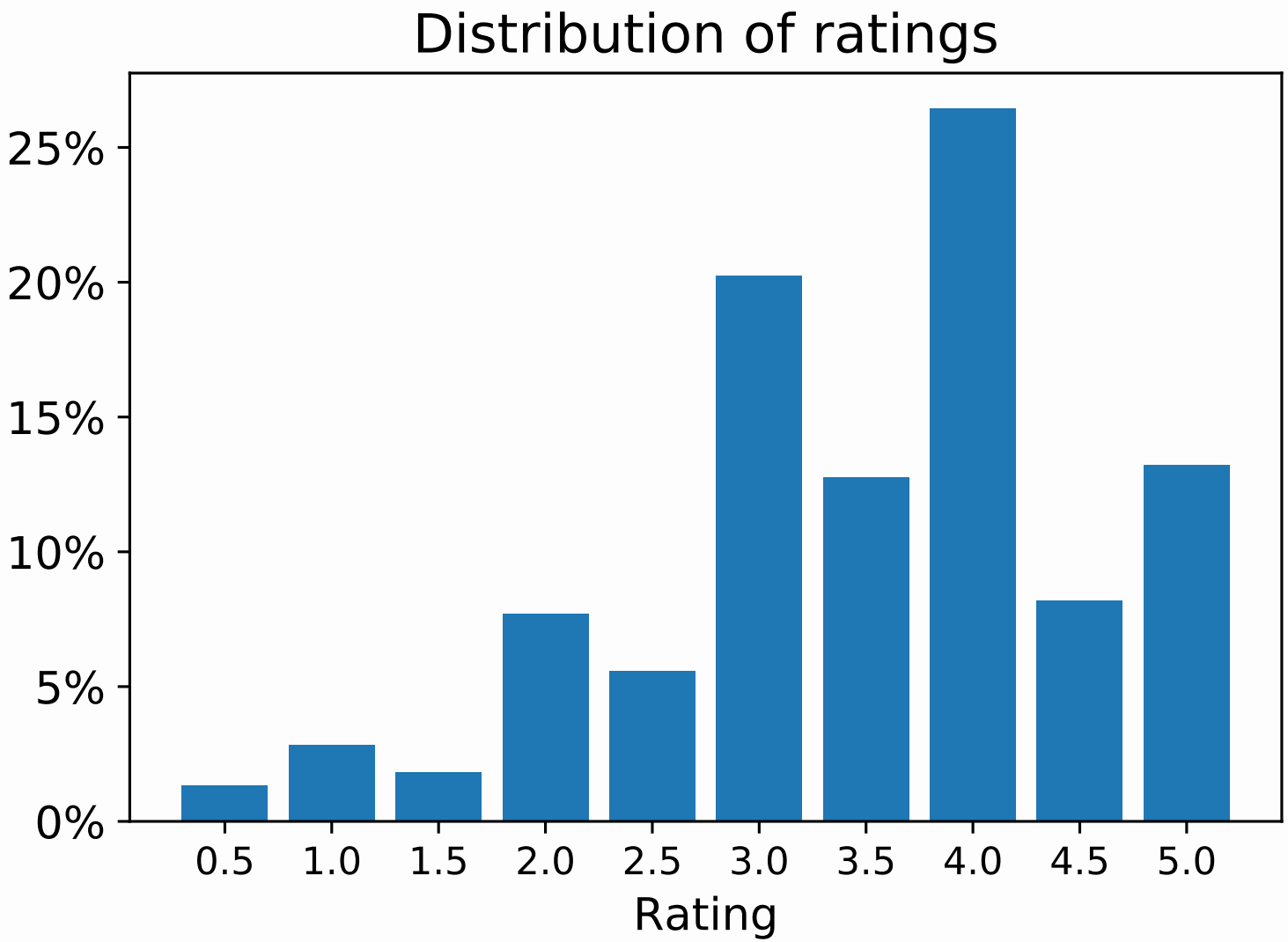
MovieID	UserID	Genres	Title	Rating
38	587	Action-Crime-Drama	Dead Presidents (1995)	3.0
38	596	Action-Crime-Drama	Dead Presidents (1995)	3.0
38	598	Action-Crime-Drama	Dead Presidents (1995)	3.0
39	5	Drama	Restoration (1995)	4.5
39	32	Drama	Restoration (1995)	2.0
39	287	Drama	Restoration (1995)	3.0

Reading the dataset cont'd

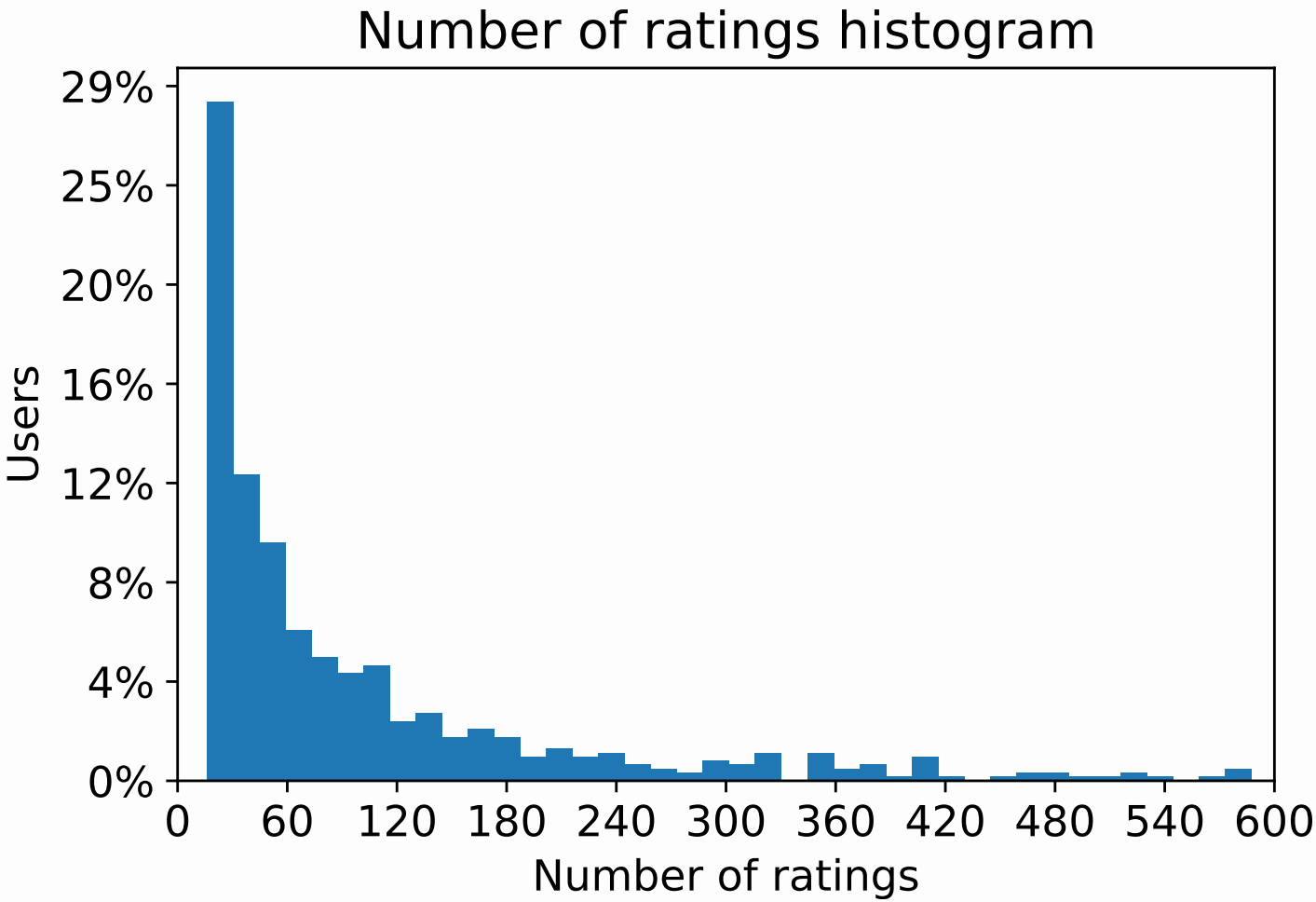
Resulting rating matrix:

```
array([[4. , 0. , 4. , ..., 0. , 0. , 0. ],
       [0. , 0. , 0. , ..., 0. , 0. , 0. ],
       [0. , 0. , 0. , ..., 0. , 0. , 0. ],
       ...,
       [2.5, 2. , 2. , ..., 0. , 0. , 0. ],
       [3. , 0. , 0. , ..., 0. , 0. , 0. ],
       [5. , 0. , 0. , ..., 0. , 0. , 0. ]])
```

Exploring the dataset



Exploring the dataset cont'd



Matrix sparsity: 1.59%.

Mathematical setting

Formally, the model is based on matrix factorisation.

- Aim: $\underset{M\times N}{R} \approx \widetilde{R} = \underset{M\times k}{X}\,\underset{k\times N}{Y}$, with k latent factors;
- Objective function:
$$\sum_{u,i} c_{ui}(r_{ui} - \mathbf{x_u^T y_i})^2 + \lambda\left(\sum_u \|\mathbf{x_u}\|^2 + \sum_i \|\mathbf{y_i}\|^2\right);$$
- Non-convex problem, solved using Weighted Alternating Least Squares;
- λ determined using CV;

Choice of weights

The choice of weights is problem dependent. Let $c_i = \sum_{u,i} 1$ if $r_{ui} > 0$.

Explicit feedback:

- $c_{ui} = \omega_0 + \omega_1 c_i$, where ω_0 weights the unobserved entries, and ω_1 the observed ones.

Implicit feedback:

- $C = 1 + \alpha R$, where $\alpha = 40$ provides good results [3].

Both choices lead to dense problems: this is preferable in order to avoid folding.

Performance concerns

Analytical solution for WALS:

$$\mathbf{x_u} = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u \mathbf{r_u}$$ AUX

Where $C^u = diag(\mathbf{c_u})$.

For n users and k latent factors, the computation is $O(k^3 + k^2 n)$ for each of the m items.

Performance concerns

Analytical solution for WALS:

$$\mathbf{x_u} = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u \mathbf{r_u} \quad \text{AUX}$$

Where $C^u = diag(\mathbf{c_u})$.

For n users and k latent factors, the computation is $O(k^3 + k^2 n)$ for each of the m items.

```
for u in range(1, M):
    Cu = diag(C[u,:])
    A = multi_dot([Y.T, Cu, Y]) + lam * eye(K)
    b = multi_dot([Y.T, Cu, R[u, :]])
    X_u = solve(A, b)
```

Note: this is embarrassingly parallel!

Cross validation

The best value for the regression coefficient λ can be determined using cross validation.

Splitting into train and test set for CV:

- Cannot trivially split the rows of R;
- Random `(user, item)` pairs should be selected;
- If $C = 1 + \alpha R$, models with different values of α cannot be compared.

Cross validation

The best value for the regression coefficient λ can be determined using cross validation.

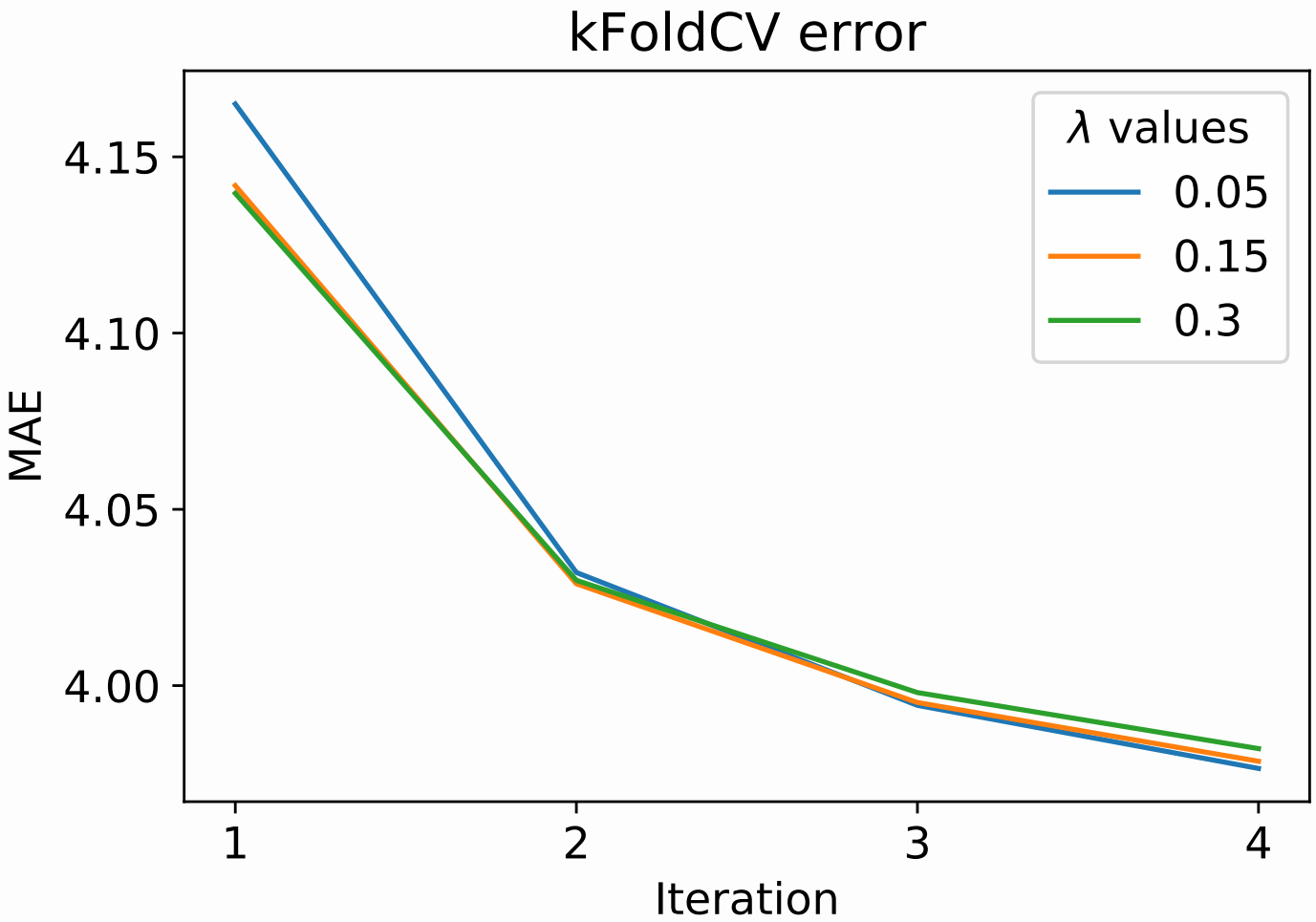
Splitting into train and test set for CV:

- Cannot trivially split the rows of R;
- Random `(user, item)` pairs should be selected;
- If $C = 1 + \alpha R$, models with different values of α cannot be compared.

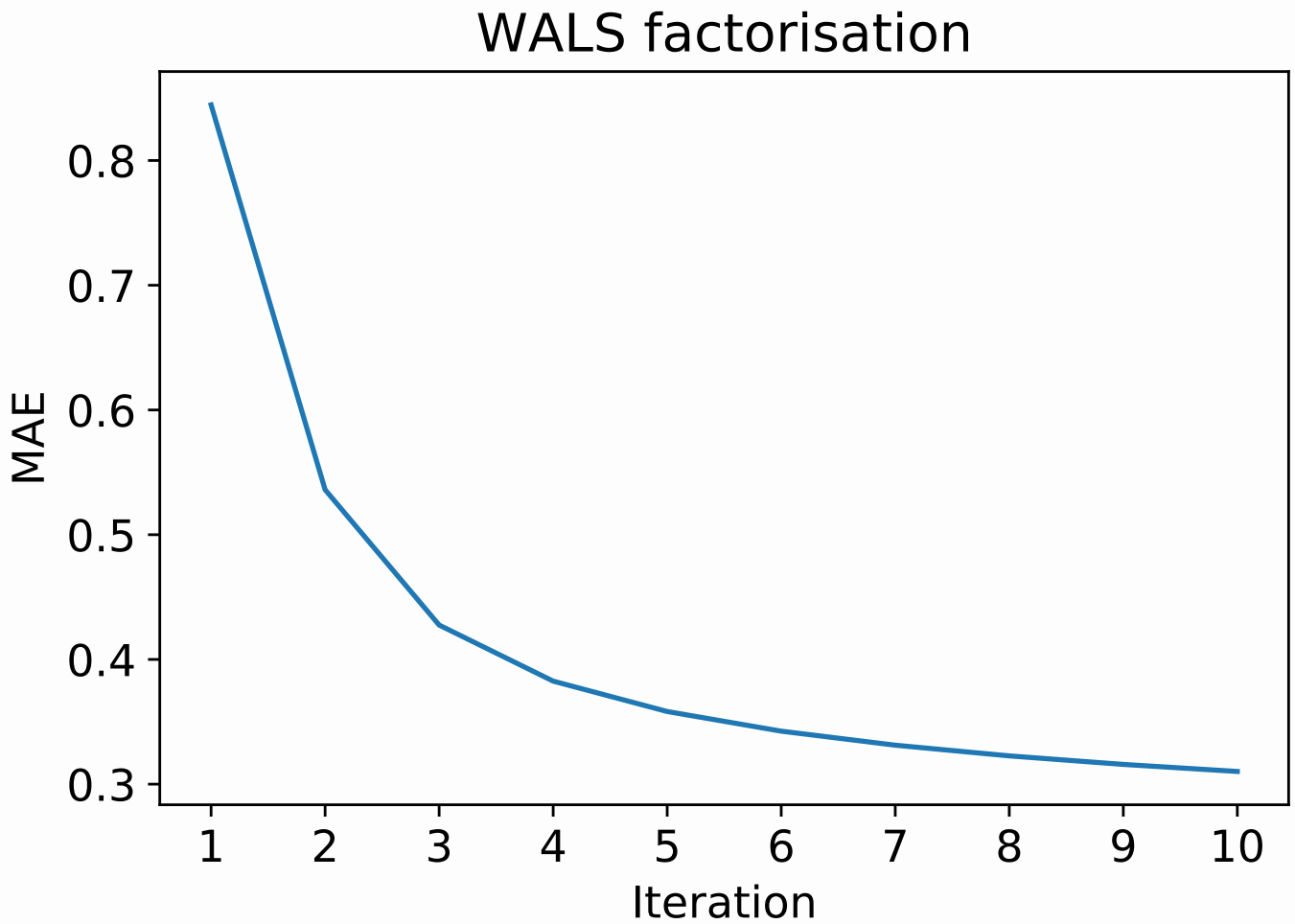
Choice of error function:

- $RMSE = \sqrt{\frac{1}{|\widetilde{R}|} \sum_{u,i \in \widetilde{R}} (r_{ui} - \widetilde{r}_{ui})^2}$
- $MAE = \frac{1}{|\widetilde{R}|} \sum_{u,i \in \widetilde{R}} |r_{ui} - \widetilde{r}_{ui}|$

Cross validation: results



Execution time: $\approx 3\,h$ for 4 iterations and $k = 100$.
What is the meaning of the high MAE?



Execution time: $\approx 52\,min$ for 10 iterations, $k = 100$ and $\lambda = 0.15$.

Recommendation results

Recommended movies for `user_id = 2`:

Pred.	Title	Genres	AVG_Rat.
6.82	Reservoir Dogs (1992)	Crime-Mystery-Thriller	4.2
6.60	Dances with Wolves (1990)	Adventure-Drama-Western	3.8
5.89	Gostbusters (1984)	Action-Comedy-Sci-Fi	3.7
5.32	American History X (1998)	Crime-Drama	4.2
4.79	Star Wars: Episode VI (1983)	Action-Adventure-Sci-Fi	4.1
4.29	Dave (1993)	Comedy-Romance	3.5
4.24	Fargo (1996)	Comedy-Crime-Drama-Thriller	4.1
4.17	Independence Day (1996)	Action-Adventure-Sci-Fi-Thriller	3.3
4.14	Young Frankenstein (1974)	Comedy-Fantasy	3.9
4.10	Angels in the Outfield (1994)	Children-Comedy	2.3

Recommendation results cont'd

Best rated movies for `user_id = 2`:

Title	Genres	Rating
Android (1982)	Sci-Fi	5.0
Escape from L.A. (1996)	Action-Adventure-Sci-Fi-Thriller	5.0
Saturn 3 (1980)	Adventure-Sci-Fi-Thriller	5.0
The Lair of the White Worm (1988)	Comedy-Horror	5.0
Road Warrior, The (Mad Max 2) (1981)	Action-Adventure-Sci-Fi-Thriller	5.0
Hangar 18 (1980)	Action-Sci-Fi-Thriller	5.0
Galaxy of Terror (Quest) (1981)	Action-Horror-Mystery-Sci-Fi	5.0
Piranha (1978)	Horror-Sci-Fi	4.5
Conan the Barbarian (1982)	Action-Adventure-Fantasy	4.5
Looker (1981)	Drama-Horror-Sci-Fi-Thriller	4.5

System evaluation

Recommender system can be evaluated using two approaches.

Offline evaluation (easier to compare algorithms):

- Low test RMSE/MAE;
- High precision/recall;
- Decent catalogue coverage.

Online evaluation:

- A/B testing (measuring Click-Through Rate and Conversion Rate).

System evaluation

Recommender system can be evaluated using two approaches.

Offline evaluation (easier to compare algorithms):

- Low test RMSE/MAE;
- High precision/recall;
- Decent catalogue coverage.

Online evaluation:

- A/B testing (measuring Click-Through Rate and Conversion Rate).

My results for system evaluation:

- For each user, the 10 most recent ratings are used for testing;
- Mean precision and recall at 10: 2.8%.

Similar items

Once the factorisation has been performed, the Y matrix contains a representation of the items using k latent factors.

Using the item embedding we can, for a given movie, suggest the most similar movies in the dataset:

- For a given `movie_id` i, compute the cosine similarity with all the other movies j as

$$\text{sim}(i, j) = \frac{\mathbf{y}_i \cdot \mathbf{y}_j}{\|\mathbf{y}_i\| \cdot \|\mathbf{y}_j\|}$$

- Return a ranked list, sorting the resulting movies by their similarity.

Note, the similarity is $\in [-1, 1]$

Similar items: results

Similar to: "The Lord of the Rings: The Fellowship of the Ring"

Title	Genres	Similarity
LOTR: The Fellowship of the Ring	Adventure-Fantasy	1.00
LOTR: The Return of the King	Action-Adventure-Drama-Fantasy	0.83
LOTR: The Two Towers	Adventure-Fantasy	0.79
The Sixth Sense	Drama-Horror-Mystery	0.78
In the Name of the Father	Drama	0.77
Twelve Monkeys	Mystery-Sci-Fi-Thriller	0.79

Similar to: "Star Wars: Episode VI - The Return of the Jedi"

Title	Genres	Similarity
Star Wars: Episode VI	Action-Adventure-Sci-Fi	1.00
Star Wars: Episode V	Action-Adventure-Sci-Fi	0.69
The Silence of the Lambs	Crime-Horror-Thriller	0.61
Independence Day	Action-Adventure-Sci-Fi-Thriller	0.58
Star Wars: Episode IV	Action-Adventure-Sci-Fi	0.55
Gladiator	Action-Adventure-Drama	0.54

Recommendation for new users

What if a new user (a set of rated movies on which the system has not been trained) asks for recommendations?

- Naive solution: add the new user to the R matrix and recompute the factorisation.

This is slow and computationally demanding.

- Better/faster solution: keep the item embedding Y fixed and compute a single iteration of the WALS algorithm.

But we need to assume that Y does not change much if a few users are added into the system. If it does change, a full factorisation should be performed.

Recommendation for new users cont'd

For a new user $M + 1$:

1. Take Y from full matrix factorisation;
2. Obtain a user representation $\mathbf{x}_{M+1}$ (a single linear system!);
3. Compute the predicted ratings $\mathbf{r}_{M+1} = \mathbf{x}_{M+1} Y$ and sort items by their score.

```
u = M + 1
Cu = diag(C[u,:])
A = multi_dot([Y.T, Cu, Y]) + lam * eye(K)
b = multi_dot([Y.T, Cu, R[u, :]])
X_u = solve(A, b)
```

New users: results

New user with 50 randomly rated movies.

Recommended movies:

Prediction	Title	Genres	AVG_Rating
7.70	Apocalypse Now	Action-Drama-War	4.2
6.76	Zodiac	Crime-Drama-Thriller	3.7
6.31	Singin' in the Rain	Comedy-Musical-Romance	4.1
6.03	ToCatcha Thief	Crime-Mystery-Romance-Thriller	4.2
5.87	Malcolm X	Drama	4.1

Best rated movies:

Title	Genres	Rating
Once Upon a Time in China	Action-Adventure-Drama	5.0
Picnic at Hanging Rock	Drama-Mystery	5.0
Kung Fu Panda: Secrets of the Masters	Animation-Children	5.0
$[REC]^2$	Horror-Thriller	5.0
In the Heat of the Night	Drama-Mystery	5.0

Cold start problem: users

What if a new user has rated too few movies or even no movie at all?

If possible, gather information:

- Old MovieLens approach: new users must rate 15 movies;
- New MovieLens approach: choose among groups of related movies;
- Use cross-domain information (i.e., book ratings);
- If personality/social network information is available, link similar users;

If no feedback and no information are available, provide popularity-based or random recommendations.

Cold start problem: users cont'd

Proposed solution:

```
Too few movies! Most poular movies will be suggested.
```

Title	Genres	AVG_Rat	Counts
Forrest Gump	Comedy-Drama-Romance-War	4.1	322
Shawshank Redemption, The	Crime-Drama	4.4	315
Pulp Fiction	Comedy-Crime-Drama-Thriller	4.1	307
Matrix, The	Action-Sci-Fi-Thriller	4.1	263
Star Wars: Episode IV	Action-Adventure-Sci-Fi	4.2	251
Braveheart	Action-Drama-War	4.0	237
Silence of the Lambs, The	Crime-Horror-Thriller	4.1	232
Jurassic Park	Action-Adventure-Sci-Fi-Thriller	3.7	219

In real cases, take care of *popularity bias*.

Cold start problem: movies

What if a movie in the dataset has few/no ratings?

This is not a problem for content-based filtering, which needs no historical information about items.

For collaborative filtering:

- Ask users for help (e.g., ExcUseMe algorithm [1]);
- Use a freshness measure.

Cold start problem: movies

What if a movie in the dataset has few/no ratings?

This is not a problem for content-based filtering, which needs no historical information about items.

For collaborative filtering:

- Ask users for help (e.g., ExcUseMe algorithm [1]);
- Use a freshness measure.

A concrete instance: computing cosine similarity with null rows in item embedding (caused by too few/no interactions for the item).

```
# Computing sim(y_i, y_j).
if len(y_i) == 0 or len(y_j):
    return -1
```

How could the current system be improved?

- Use a larger dataset;
- Parallelise matrix factorisation;
- Implement a freshness measure;
- Include a static score (rating) or freshness score in computing similarity;
- Include a diversity measure (aiming at serendipity).

Conclusions

Conclusions.

References

[1] M. Aharon, O. Anava, N. Avigdor-Elgrabli, D. Drachsler-Cohen, S. Golan, and O. Somekh.

Excuseme: Asking users to help in item cold-start recommendations.

In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 83–90. ACM, 2015.

[2] F. M. Harper and J. A. Konstan.

The movielens datasets: History and context.

Acm transactions on interactive intelligent systems (tiis), 5(4):19, 2016.

[3] Y. Hu, Y. Koren, and C. Volinsky.

Collaborative filtering for implicit feedback datasets.

In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[4] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan.

Large-scale parallel collaborative filtering for the netflix prize.

In *International conference on algorithmic applications in management*, pages 337–348. Springer, 2008.

Auxiliary slides

WALS: ANALYTIC SOLUTION

Starting from the objective function

$$\mathcal{L} = \sum_{u,i} c_{ui}(r_{ui} - \mathbf{x_u^T y_i})^2 + \lambda \left(\sum_u \|\mathbf{x_u}\|^2 + \sum_i \|\mathbf{y_i}\|^2 \right):$$

- $\frac{\partial \mathcal{L}}{\partial \mathbf{x_u}} = -2 \sum_i c_{ui}(r_{ui} - \mathbf{x_u^T y_i})\mathbf{y_i} + 2\lambda \mathbf{x_u} = 0$
- Scalar product commutativity:

$$-2 \sum_i c_{ui}(r_{ui} - \mathbf{y_i^T x_u})\mathbf{y_i} + 2\lambda \mathbf{x_u} = 0$$

- Using the definition of matrix product:

$$-2 Y^T C^u \mathbf{r_u} + 2 Y^T C^u Y \mathbf{x_u} + 2\lambda \mathbf{x_u} = 0$$

- Rearranging:

$$\mathbf{x_u} = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u \mathbf{r_u}$$