The Netflix Prize and Collaborative Filtering

March 8, 2018

Outline

Netflix Prize

Collaborative Filtering

Identifying Idiosyncratic Raters

Outline

Netflix Prize

Collaborative Filtering

Identifying Idiosyncratic Raters

Netflix Prize

- Netflix users rate movies 1–5 stars.
- Netflix wants to recommend movies to users that they will like.

Goal: Predict rating that user will give movie they haven't seen yet.

Netflix Challenge: Data

	Titanic	Harry Potter	Indiana Jones	The Room
Josephine	5		3	1
Thomas			5	1
Sophia	5	4	3	1
Pratik				1
Mark		2		1

Predict the rating Josephine will give Harry Potter:

- Simple Idea: Predict 3 because Harry Potter received an average of 3.
- Collaborative Filtering Idea: Predict 4 because Josephine and Sophia have similar tastes and Sophia gave HP a 4.

Evaluation Criteria

- Hide red cells when training the algorithm:

	Titanic	Harry Potter	Indiana Jones	The Room
Josephine	5		3	1
Thomas			5	1
Sophia	5	4	3	1
Pratik				1
Mark		2		1

- Algorithm predicts $\widehat{s}_k$ for cell s_k. (every red cell)
- $RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(\widehat{s}_k - s_k)^2}$. (could use other criteria)

Netflix Challenge: Data

Data Summary:

- $p =$ number of movies $\approx 20,000$
- $n =$ number of users $\approx 500,000$
- 100 million ratings in training set
- 2 million ratings in test set

Outline

Netflix Prize

Collaborative Filtering

Identifying Idiosyncratic Raters

Simple Models

1. For user i–movie j predict 3 stars. (RMSE ≤ 2)
   - Does not use any information in training.
2. $\widehat{\mu} =$ mean stars in training. For user i–movie j, predict $\widehat{\mu}$.
   - Does not use any information about movie j.
3. $\widehat{\mu}_j =$ mean training stars for movie j. For user i–movie j, predict $\widehat{\mu}_j$.
   - Does not use any information about user i.

Note: Method 3 is an average of responses. Let

$$R_j = \text{ all users who rated movie } j.$$

Then,

$$\text{prediction for user i } = \frac{1}{\# R_j} \sum_{k \in R_j} x_{kj}$$

Weighted Average

Idea: Weight average by how close users i and k are to each other.

- Let w_{ik} be a measure of closeness (based on ratings) of i and k.
- Then

$$\text{prediction for user i } = \frac{1}{\sum_{k \in R_j} w_{ik}} \sum_{k \in R_j} w_{ik} x_{kj}$$

Result: The same movie will receive a different prediction for different users.

Example

	Titanic	Harry Potter	Indiana Jones	The Room
Josephine	5		3	1
Thomas			5	1
Sophia	5	4	3	1
Pratik				1
Mark		2		1

Predict **Harry Potter** rating for **Josephine**. Suppose:

- Josephine and Sophia have $w = 1$
- Josephine and Mark have $w = 1/2$

$$\text{prediction} = \frac{4 * 1 + 2 * (1/2)}{1 + 1/2} = 3.33$$

Notes

- Many different possible ways to measure similarity
  - `http://www.dataperspective.info/2014/05/basic-recommendation-engine-using-r.html`
- Methods which build similarities between users are "User Based" Collaborative Filtering
- "Item Based" Collaborative Filtering constructs similarities between movies.
  - Terminator and Die Hard are similar because users give them similar ratings.

Outline

Netflix Prize

Collaborative Filtering

Identifying Idiosyncratic Raters

Not all Raters are Useful

Reasons for unusual ratings:

- Some users assign a random number of stars just to get to the next screen.
- Robots / trolls may deliberately give confusing ratings to movies.

Goal:

- Identify these users as a cleaning step before using a collaborative filtering algorithm.

Formal Statistical Model

Problem 24 from Lange Chapter 13:

- Suppose there are 5 possible ratings.
- User i operates in consensus mode $1 - \pi_i$ fraction of time.
  - In consensue mode i rates j with distribution $(c_{j1}, c_{j2}, c_{j3}, c_{j4}, c_{j5})$
- User i operates in quirky mode π_i fraction of time.
  - In quirky mode i has private rating distribution $(q_{i1}, q_{i2}, q_{i3}, q_{i4}, q_{i5})$
- The larger π_i, the more unusual the user.

The likelihood is

$$L = \prod_{i=1}^{n} \prod_{j \in M_i} (\pi_i q_{ix_{ij}} + (1 - \pi_i) c_{jx_{ij}})$$

where M_i is all movies rated by user i.

We study how to optimize this likelihood with the EM algorithm next week.

Software

- Python
  - Scikit for Recommender systems
    https://github.com/NicolasHug/Surprise
  - Example with MovieLens dataset: http://blog.ethanrosenthal.com/2015/11/02/intro-to-collaborative-filtering/
- R
  - R package:
    https://CRAN.R-project.org/package=recommenderlab
  - useage case: https://rpubs.com/jt_rpubs/285729
  - description of collaborative filtering
    https://www.smartcat.io/blog/2017/improved-r-implementation-of-collaborative-filtering/