Configuring User-User Collaborative Filtering

Introduction

- Previous lectures
  - User-user collaborative filtering
  - How user-user CF works
- This lecture
  - Customizations and design decisions

Overview

- Selecting Neighborhoods
- Scoring Items from Neighborhoods
- Normalizing Data
- Computing Similarities
  - Algorithms
  - Tweaks
- Good Baseline Configuration

Overview

- **Selecting Neighborhoods**
- Scoring Items from Neighborhoods
- Normalizing Data
- Computing Similarities
  - Algorithms
  - Tweaks
- Good Baseline Configuration

Selecting Neighborhoods

- All the neighbors
- Threshold similarity or distance
- Random neighbors
- Top-*N* neighbors by similarity or distance
- Neighbors in a cluster

How Many Neighbors?

- In theory, the more the better
  - If you have a good similarity metric
- In practice, noise from dissimilar neighbors decreases usefulness
- Between 25 and 100 is often used
  - 30–50 often good for movies

Overview

- Selecting Neighborhoods
- **Scoring Items from Neighborhoods**
- Normalizing Data
- Computing Similarities
  - Algorithms
  - Tweaks
- Good Baseline Configuration

Scoring from Neighborhoods

- Average
- Weighted average
- Multiple linear regression

Weighted average is common, simple, and works well

Overview

- Selecting Neighborhoods
- Scoring Items from Neighborhoods
- **Normalizing Data**
- Computing Similarities
  - Algorithms
  - Tweaks
- Good Baseline Configuration

What's wrong with data?

- Users rate differently
- Some rate high, others low
- Some use more of the scale than others
- Averaging ignores these differences
- Normalization compensates for them

Common Normalizations

- Subtract user mean rating
- Convert to *z*-score (1 = 1 standard deviation above mean)
- Subtract item or item-user mean

Must reverse normalization after computing

$$s(u,i) = \frac{\sum_v w_{uv}(r_{vi} - \bar{r}_v)}{\sum_v |w_{uv}|} + \bar{r}_u$$

Overview

- Selecting Neighborhoods
- Scoring Items from Neighborhoods
- Normalizing Data
- **Computing Similarities**
  - Algorithms
  - Tweaks
- Good Baseline Configuration

Computing Similarities

- Last time: Pearson correlation

$$sim(u,v) = \frac{\sum(r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum(r_{ui} - \bar{r}_u)^2}\sqrt{\sum(r_{vi} - \bar{r}_v)^2}}$$

- Usually only over ratings in common
- User normalization not needed
- Spearman rank correlation is Pearson applied to *ranks*
- Hasn't been found to work as well

Problem: what about little data?

- Suppose users have 1 rating in common
- Pearson correlation is 1
- Are the users really similar?

Weighting Similarity

- Compute sums in denominator over all of each user's ratings
- Result: users with few ratings in common, but many ratings, will have lower similarity
- Similar to *significance weighting* in older literature
- Equivalent to *cosine similarity* over mean-centered user rating vectors

Overview

- Selecting Neighborhoods
- Scoring Items from Neighborhoods
- Normalizing Data
- Computing Similarities
  - Algorithms
  - Tweaks
- **Good Baseline Configuration**

Good Baseline Configuration

- Top *N* neighbors (~30)
- Weighted averaging
- User-mean or z-score normalization
- Cosine similarity over normalized ratings

Conclusion

- There are a variety of configuration points
- Current research has suggested some that work well
- Next course will discuss evaluation methods you can use to find good options for your application