4-3: User-User Variations and Tuning

Introduction to Recommender Systems

Introduction

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

Learning Objectives

- Know the main implementation decisions to make in user-user CF
- Understand different options and why they might be selected
- Have a best-practice starting point for configuring a user-user CF recommender

Overview

- Selecting Neighborhoods
- Scoring Items from Neighborhoods
- Normalizing Data
- Computing Similarities
  - Algorithms
  - Tweaks
- Additional Options

Overview

- **Selecting Neighborhoods**
- Scoring Items from Neighborhoods
- Normalizing Data
- Computing Similarities
  - Algorithms
  - Tweaks
- Additional Options

Selecting Neighborhoods

- All the neighbors
- Threshold similarity or distance
- Random neighbors
- Top-*N* neighbors by similarity or distance

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
- Fewer neighbors → lower coverage

Overview

- Selecting Neighborhoods
- **Scoring Items from Neighborhoods**
- Normalizing Data
- Computing Similarities
  - Algorithms
  - Tweaks
- Additional Options

Scoring Items 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
- Additional Options

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

Mean-centering

- Subtract user mean prior to computing
- Re-add when needed

z-score normalization

- Mean-center, and divide by standard deviation
- Normalizes for the spread across the scale
- Small additional gain in prediction accuracy over mean-centering

Other normalizations

- Subtract item mean
- Subtract item-user mean

Overview

- Selecting Neighborhoods
- Scoring Items from Neighborhoods
- Normalizing Data
- **Computing Similarities**
  - Algorithms
  - Tweaks
- Additional Options

Computing Similarities

- Last time: Pearson correlation

$$s(a,u)\frac{\sum(r_{ai}-\mu_a)(r_{ui}-\mu_u)}{\sqrt{\sum(r_{ai}-\mu_a)^2}\sqrt{\sum(r_{ui}-\mu_u)^2}}$$

- 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?

Solution: significance weighting

- Weight similarity by confidence
- Simple approach: multiply by 1/min(n,50)
  - $<$ 50 common ratings: scaled down by # of common ratings
  - $\geq$ 50 common ratings: unscaled
- Can also do Bayesian damping

Vector Similarity

- Compute cosine of user vectors in rating space

$$sim(a,u) = \frac{\mathbf{a} \cdot \mathbf{u}}{|\mathbf{a}|\,|\mathbf{u}|} = \frac{\sum r_{ai} r_{ui}}{\sqrt{\sum r_{ai}^2}\sqrt{\sum r_{ui}^2}}$$

- With user-mean norm: Pearson correlation!

Self-weighting Similarity

- Cosine has built-in significance weighting
- Weights proportionally to ratio of common ratings & total ratings (roughly)
- Similar effect as using overall σ_u instead of just over common ratings in Pearson

Overview

- Selecting Neighborhoods
- Scoring Items from Neighborhoods
- Normalizing Data
- Computing Similarities
  - Algorithms
  - Tweaks
- **Additional Options**

Clustering

- Cluster users
- Pick user's cluster to generate predictions
- Doesn't work particularly well

Pre-computation

- Expensive
- Users move as their ratings change

Baseline Configuration

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

Conclusion

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

4-3: User-User Variations and Tuning

Introduction to Recommender Systems

$$sim(a,u) = \frac{\vec{a}\cdot\vec{u}}{\|\vec{a}\|\|\vec{u}\|}$$

$$= \frac{\sum_i \hat{r}_{ai}\cdot\hat{r}_{ui}}{\sqrt{\sum_i \hat{r}_{ai}^2}\sqrt{\sum_i \hat{r}_{ui}^2}} \qquad \hat{r}_{ai} = r_{ai} - \mu_a$$

$$= \frac{\sum (r_{ai}-\mu_a)(r_{ui}-\mu_u)}{\sqrt{\sum (r_{ai}-\mu_a)^2}\sqrt{\sum (r_{ui}-\mu_u)^2}}$$

Scale By $\sim \dfrac{|R_a \cap R_u|}{|R_a||R_u|}$

$$sim(a,u) = \frac{\sum \overbrace{(r_{ai}-\mu_a)}^{x}\overbrace{(r_{ui}-\mu_u)}^{y}}{\sqrt{\sum (r_{ai}-\mu_a)^2}\sqrt{\sum (r_{ui}-\mu_u)^2} + K}$$

$$\frac{xy}{\sqrt{x^2}\sqrt{y^2}} = 1$$

$$sim(a,u) \cdot \frac{1}{\min(50, |R_a \cap R_u|)}$$

$$\mu + \hat{\mu}_i + \hat{\mu}_u$$

$$\hat{\mu}_i = \frac{\sum_u (r_{ui} - \mu)}{\#\text{of ratings }(i)}$$

$$\hat{\mu}_u = \frac{\sum_i (r_{ui} - \mu - \hat{\mu}_i)}{\#(u)}$$

$$s(a,i) = \frac{\sum_{u \in R} r_{ui} \cdot sim(a,u)}{\sum_u sim(a,u)}$$

$$\frac{\sum_u (r_{ui} - \mu_u) \cdot sim(a,u)}{\sum_u sim(a,u)} + \mu_a$$

$$z_{ui} = \frac{r_{ui} - \mu_u}{\sigma_u}$$

$$s(a,i) = \frac{\sum_u z_{ui} \cdot sim(a,u)}{\sum_u |sim(a,u)|} \cdot \sigma_a + \mu_a$$

$$r \quad \frac{\sum (r_{ai} - \mu_{ai})(r_{ui} - \mu_u)}{\sqrt{\sum (r_{ai} - \mu_a)^2} \sqrt{\sum (r_{ui} - \mu_u)^2}}$$

$$\frac{\sum (r_{ai} - \mu_a)(r_{ui} - \mu_u)}{\sqrt{\sum (r_{ai} - \mu_a)^2} \sqrt{\sum (r_{ui} - \mu_u)^2}}$$