6-2: Item-Item Algorithm

Introduction to Recommender Systems

Introduction

- We're now into the 2nd major personalized algorithm: item-item CF
- This lecture will discuss the algorithm in more detail
- Also: design space and performance implications

Structure of Item-Item CF

- Pre-compute item similarities over all pairs of items
- Look for items similar to those the user likes
  - Or has purchased
  - Or has in their basket

Components

- Item similarity function
- Model builder
- Neighborhood selection strategy
- Item score aggregation function

Item Similarities

- Usually use cosine similarity between item rating vectors
- Often normalize user ratings first
  - Subtract user mean
  - Subtract item mean

$$sim(i,j) = \frac{\sum_{u \in U(i) \cap U(j)} \hat{r}_{ui}\hat{r}_{uj}}{\sqrt{\sum_u \hat{r}_{ui}^2}\sqrt{\sum_u \hat{r}_{uj}^2}}$$

Scoring Items

- Score is driven by item
- For each item to score:
  - Find similar items the user has rated
  - Compute weighted average of user's ratings

$$p_{ui} = \frac{\sum_{j \in N} sim(i,j) r_{uj}}{\sum_{j \in N} |sim(i,j)|}$$

Picking Neighbors

- Score formula had a neighborhood *N*
- Neighbors are usually *k* most similar items
  - That the user has rated
- Good value of *k* important
  - *k* too small → inaccurate scores
  - *k* too large → too much noise (low-similarity items)
  - *k*=20 often works well

Building the Model

- Pre-compute similarities for all pairs of items
  - Item stability makes similarity pre-computation feasible
- Naïvely: $O(|I|)^2$
  - If symmetric: only need to compute one direction
  - Sometimes can skip pairs

Truncating the Model

- Don't need to keep the whole I^2 model
- But need enough neighbors to find neighbors at score time
  - Since user hasn't rated everything, need $M >> k$ neighbors per item in model
- Balance memory use with accuracy and coverage
  - Mild runtime impact, if neighbors are sorted

Tuning the model

- Tune using cross-validation
- Need to find good values for
  - baseline and normalization
  - similarity function (or just use cosine)
  - neighborhood size k
  - model size M
  - sometimes similarity is damped, but often doesn't help much

Conclusion

- Item-item is efficient and straightforward
- A few parameters need tuning for specific data, domain
- Next: more tweaks
  - applying to unary data (implicit feedback)
  - repurposing and hybridization

6-2: Item-Item Algorithm

Introduction to Recommender Systems