Item-Item Algorithm

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

Previously…

We saw the *user-user* algorithm:

$$s(u,i) = \frac{\sum_{v \in V} w_{uv}(r_{vi} - \bar{r}_v)}{\sum_{v \in V} w_{uv}} + \bar{r}_u$$

We can also average over items!

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

$$s(u,i) = \frac{\sum_v w_{uv} r_{vi}}{\sum_v |w_{uv}|}$$

$$s(u,i) = \frac{\sum_{j \in N} w_{ij} r_{uj}}{\sum_j |w_{ij}|}$$

$w_{ij} = ?$

$$= sim(i,j) = \cos(\hat{r}_i, \hat{r}_j) = \frac{\hat{r}_i \cdot \hat{r}_j}{\|\hat{r}_i\|_2 \|\hat{r}_j\|_2}$$

$$= \frac{\sum_u \hat{r}_{ui} \hat{r}_{uj}}{\sqrt{\sum_u \hat{r}_{ui}^2} \sqrt{\sum_u \hat{r}_{uj}^2}} = \frac{\sum_u (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_u (r_{ui} - \bar{r}_i)^2} \sqrt{\sum_u (r_{uj} - \bar{r}_j)^2}}$$

How to pick neighbors?

$N(i;u)$ k most similar items to i that u has rated

$$s(u,i) = \frac{\sum_{j \in N(i;u)} w_{ij} (r_{uj} - \bar{r}_j)}{\sum_{j \in N(i;u)} |w_{ij}|} + \bar{r}_i$$

Components

- Item similarity function
  - Also called *interpolation weight* w_{ij}
- Model builder
- Neighborhood selection strategy
- Item score aggregation function

Item Similarities

- Usually use cosine similarity between item rating vectors
- Normalize user ratings first
  - Subtract item mean
  - Historically: subtract user mean
  - Treat missing values as 0

$$w_{ij} = \text{sim}(i,j) = \frac{\sum_{u \in U_i \cap U_j} \hat{r}_{ui} \hat{r}_{uj}}{\sqrt{\sum \hat{r}_{uj}} \sqrt{\sum \hat{r}_{uj}}}$$

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

$$s(i;u) = \frac{\sum_{j \in N(i;u)} w_{ij}(r_{uj} - \bar{r}_j)}{\sum_{j \in N(i;u)} |w_{ij}|} + \bar{r}_i$$

  - Average normalized ratings, denormalize when done
  - Can consider other methods like linear regression

Picking Neighbors

- Score formula has neighborhood $N(i; u)$
- Neighbors are usually k most similar items
  - That the user has rated (hence u)
- Good value of k important
  - k too small $\rightarrow$ inaccurate scores
  - k too large $\rightarrow$ too much noise
  - $k = 20$ often works well

Building the Model

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

Truncating the Model

- Don't need to keep the whole I^2 model
- Need enough neighbors to find neighbors at score time
  - Since user hasn't rated everything, need $M \gg k$ neighbors per item in model
- Can drop nonpositive neighbors
- Balance memory use with accuracy and coverage
  - Mild runtime improvements as well

Model Building Algorithm

initialize W to empty weight matrix

for $i \in I$

  for $j \in I$ s.t. $i \neq j$

    if $\text{sim}(i, j) > 0$

      $w_{ij} \leftarrow \text{sim}(i, j)$

  clear all but M largest values of row $\vec{w}_i$

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

Item-Item Algorithm