Collaborative Filtering

- Recommend new items liked by other users similar to this user
- need items already rated by user ***and other users***
- don't need characteristics of items
  - each rating by individual user becomes characteristic
- Can combine with item characteristics
  - hybrid content/collaborative

1

Major method types

- Nearest neighbor
  - Use similarity function
  - Prediction based on previously rated items
- Matrix Factorization
  - "Latent factors"
  - Matrix decomposition
- Both use (user × item) matrix
  - vector similarity

2

Example of nearest neighbor: Preliminaries

- Notation
  - $r(u,i)$ = rating of i^{th} item by user u
  - I_u = set of items rated by user u
  - $I_{u,v}$ = set of items rated by both users u and v
  - $U_{i,j}$ = set of users that rated items i and j
- Adjust scales for user differences
  - Use average rating by user u:

    $r_u^{avg} = (1/|I_u|) * \sum_{i \text{ in } I_u} r(u,i)$
  - Adjusted ratings: $r_{adj}(u,i) = r(u,i) - r_u^{avg}$

3

One choice of similarity function: User Similarities

- similarity between users u and v
  - Pearson correlation coefficient

$$sim(u,v) = \frac{\sum_{i \text{ in } I_{u,v}} (r_{adj}(u,i)*r_{adj}(v,i))}{\left(\sum_{i \text{ in } I_{u,v}} (r_{adj}(u,i))^2 * \sum_{i \text{ in } I_{u,v}} (r_{adj}(v,i))^2 \right)^{1/2}}$$

4

Predicting User's rating of new item: User-based

For item i not rated by user u

$$r^{pred}(u,i) = r_u^{avg} + \frac{\sum_{v \text{ in } S} (sim(u,v) * r_{adj}(v, i))}{\sum_{v \text{ in } S} |sim(u,v)|}$$

S can be all users who have rated i or just those users *most similar* to u

Collaborative filtering example

user ratings

	book 1	book 2	book 3	book 4
user 1	5	1	2	0
user 2	x	5	2	5
user 3	3	1	x	2
user 4	4	0	2	?

adj. user ratings

	book 1	book 2	book 3	book 4
user 1	3	-1	0	-2
user 2	x	1	-2	1
user 3	1	-1	x	0
user 4	2	-2	0	?

Collaborative filtering example

- sim(u1,u4) = $(6+2)/(10*8)^{1/2}$ = .894
- sim(u2,u4) = $(-2)/(5*4)^{1/2}$ = -.447
- sim(u3,u4) = $(2+2)/(2*8)^{1/2}$ = 1

- predict r(u4, book4) = $2 + \frac{(-2)*.894 + 1*(-.447) + 0*1}{.894 + .447 + 1}$

$= 2 - .955 \approx 1$

Another choice of similarity function: Item Similarities

- similarity between items i and j
  - vector of ratings of users in $U_{i,j}$
  - cosine measure using adjusted ratings

$$sim(i,j) = \frac{\sum_{u \text{ in } U_{i,j}} (r_{adj}(u,i) * r_{adj}(u, j))}{\left(\sum_{u \text{ in } U_{i,j}} (r_{adj}(u,i))^2 \sum_{u \text{ in } U_{i,j}} (r_{adj}(u, j))^2 \right)^{1/2}}$$

Predicting User's rating of new item: Item-based

For item i not rated by user u

$$r^{\text{item-pred}}(u,i) = \frac{\sum_{j \text{ in } T} (\text{sim}(i,j)*r(u, j))}{\sum_{j \text{ in } T} |\text{sim}(i,j)|}$$

T can be all items in I_u or just items *most similar* to i

➢ Prediction uses only u's ratings, but similarity uses other users' ratings

9

Limitations

- May not have enough ratings for new users
- New items may not be rated by enough users
- Need "critical mass" of users
  - All similarities based on user ratings

But can take user "out of comfort zone"

10

Applying nearest-neighbor collab. filtering concepts to search

- Collaborative histories
  - How determine user similarity?
    - Clicking URL = buying product?
    - Behavior on only identical searches?
    - Exact URLs or general topic interests?
      - Hybrid content-based and behavior-based
    - Computational expense?
      - Argues for general topic-interest characterizations
  - How apply similarity?
    - Same search? or Same topic of search?
    - Bias ranking? or Bias topics of results?

11

Where are we?

- Refinement/Personalization of results
- Study techniques of

  Recommender systems
  - Content filtering
  - Collaborative filtering
    - Nearest neighbor methods
    - ➢ Matrix factorization methods

12

Matrix factorization motivation

- Matrix representation
  - users X items
  - documents X terms
- Discover/use latent factors
  - attributes, topics, features
- Factor matrices to uncover latent factors
- Don't know what latent factors represent
  - can conjecture
- For recommenders, matrix has holes
  - use factorization to fill in

13

Matrix factorization for Collaborative Filtering

- Give ratings matrix R: M users X N items
  - R has holes- R_{ij} with no value
- Want to fill in holes => predict ratings
- Idea: decompose R:

$$R=PQ^T$$

  - P is M X f; Q is N X f
  - f dimensions are latent factors
    - no interpretation but can add one
  - must choose f

14

How does decomposition help?

- estimate P and Q, leaving no holes
- get estimate of R as $R_f = PQ^T$
  - R_f has holes of R filled in
- Several methods for estimation, e.g.
  - Gradient descent
  - Stochastic gradient descent
    - Koren et al. *Matrix Factorization Techniques for Recommender Systems*, IEEE Computer, Aug 2009
  - Least squares based calculations
    - Bell et al *Modeling Relat'ships at Multiple Scales to Improve Accuracy of Large Recom. Sys.*, KDD Aug 2007.

15

Optimization

- Minimize least squares error:

err(P,Q) is defined as

$$\sum_{(u,i) \text{ in } F}(R_{(u,i)} - (PQ^T)_{(u,i)})^2$$

for F the set of (u,i) for which $R_{(u,i)}$ has a value

16

Simple Step: Gradient Descent

- Minimize for one element change:
  - choose one element of P or one element of Q to vary, say $P_{(r,s)}$
    $(PQ^T)_{(r,j)} = (\sum_{k,\, k\neq s}) P_{(r,k)} * Q_{(j,k)}) + x * Q_{(j,s)}$
  - err(P,Q) becomes equation with one unknown
    - look at only terms involving x
    - get sum over j for which $R_{(r,j)}$ has a value of:

$$(R_{(r,j)} - (PQ^T)_{(r,j)})^2 = (R_{(r,j)} - (\sum_{k,\, k\neq s}) P_{(r,k)} * Q_{(j,k)}) - x * Q_{(j,s)})^2$$

  - take derivative wrt x, set to 0, solve

17

Update step

Solution:

$$x = \frac{\sum_{j \text{ in } F} Q_{(j,s)} (R_{(r,j)} - \sum_{k\neq s} P_{(r,k)} * Q_{(j,k)})}{\sum_j Q^2_{(j,s)}}$$

For F the set of (r,j) for which $R_{(r,j)}$ has a value

- Similar equation if set element of Q to unknown y
- Iterate through elements of P, Q, repeatedly
- Find local minimum
  - improvement threshhold
- Need initial values P, Q

18

Initializing

- Many strategies for initializing
- Example:
  - fill in each hole with average of column (item) values
  - decompose using SVD to get a rank f approximation $(U'_f\, \Sigma'_f\, V'^T_f)$
  - let $P_{init} = U'_f (\Sigma'_f)^{1/2}$
  - let $Q_{init} = V'_f (\Sigma'_f)^{1/2}$
  - note this particular initialization eliminates holes

19

Matrix factorization: summary

- Very effective method
- Issues:
  - Iteration is costly
    - Wait for local optimum?
  - Must choose initial values
- Subject of ongoing research

20

High-level issues for Collaborative Filtering:

Global effects

Effects over many or all of ratings

✓ different users have different rating scales
- metadata (attributes) for items and/or users
  hybrid content/collaborative
- date of rating
- trend of user's ratings over time
- trend of item's ratings over time

Reference: Scalable Collaborative Filtering w/ Jointly Derived Neighborhood Interpolation Weights, Bell and Koren, *IEEE Intern. Conf. Data Mining* (part of winning Netflix contest team)

21

Final thought

All techniques we've seen behavior or topic oriented

What about links? What about PageRank?

22

Refining PageRank

$$\boldsymbol{pr} = (\alpha/n, \alpha/n, \dots \alpha/n)^T + (1-\alpha)\, L^T \boldsymbol{pr}$$

- let $\boldsymbol{v} = (1/n, 1/n, \dots 1/n)$
- rewrite $\boldsymbol{pr} = (\alpha)\boldsymbol{v}^T + (1-\alpha)\, L^T \boldsymbol{pr}$
- Refinement choices
  – change $\boldsymbol{v}$
  – change $\boldsymbol{L}$

23

"Topic Sensitive" PageRank

Haveliwala

- Use pre-defined topics
  – Open Directory Project (DMOZ)
    - "the largest, most comprehensive human-edited directory of the Web."
    - 16 top-level topics
- Each page has PageRank for each topic
  – Degree to which page is part of topic
- Calculate similarity of query to each topic
  – Use linear combination of topic PageRanks based on similarity values query to topic

24

Creating PageRank for a Topic

- Set C_j contains all the URLS for j^{th} topic
- change PageRank equation:

$$pr_{new}(k) = \alpha(\mathbf{1/n}) + (1-\alpha)\sum_{i \text{ with edge from } i \text{ to } k} (pr(i) / t_i)$$

$$\Downarrow$$

$$pr_{new}(k) = \alpha(\mathbf{c_k}) + (1-\alpha)\sum_{i \text{ with edge from } i \text{ to } k} (pr(i) / t_i)$$

Where $c_k = 1/|C_j|$ if the k^{th} node represents a URL in C_j
0 otherwise

- removes random jumps to nodes outside the topic

Personalized PageRank

Kamvar et. al.

- Random leaps are biased by personal interests – change ***v***
- Combined with use of block structure to make more efficient:
  - Divide Web graph into blocks (clusters)
    - Use high-level domains (e.g. princeton.edu)
  - Calc. local PageRank within each block
  - Collapse each block into 1 node – new graph
    - Weighted edges between nodes
  - Calc. PageRank with biased leaps for block structure
  - Weight local PageRanks with block PageRank
    - Use to initialize power calculation

Refinement & Personalization Summary

- Looked at several techniques to modify search
- explicit user feedback
- user behavior: history
  - user history
  - crowd history
  - collaborative history: “people like you”
- role of social networks
  - general analysis
  - relationships
- models of recommender systems