COMP9727: Recommender Systems

Lecture 3: Collaborative Filtering

Wayne Wobcke

e-mail:w.wobcke@unsw.edu.au

This Lecture

Focus on ratings prediction

- Neighbourhood Methods
  - Item-Based Similarity
  - User-Based Similarity
- Matrix Factorization
- Hybrid Recommender Systems

But recommendation requires more than (just) predicting ratings!

Collaborative Filtering

Useful when items cannot easily be categorized

- Item-based: Recommend items "similar" to those consumed
  - Amazon: Items are "similar" if users bought them together
  - Users who bought this also bought that
- User-based: Recommend items consumed by "similar" users
  - Users are "similar" if they rated the same items similarly
  - You might like this other item liked by those users

Data sources: Explicit ratings, user–system interactions

Many ways to define "similarity" of users/items

Amazon.com Item-Based Similarity

- Consider items purchased together in a transaction ("basket")
- A given item is included in many such baskets
- Items are "similar" if they have been bought together at least once
- Calculate similarity $sim(i, j)$ of items i and j
  - Each item has a set/vector of users who bought that item
  - Use Jaccard, cosine similarity, etc., on those sets/vectors
  - Rank items j in order of $sim(i, j)$
- Recommend N items $j \neq i$ similar to i, for which N?

Problems with (Movie) Ratings

- Meaning of scale changes over time: “uberization”
- Individual differences in rating interpretation
- Sparsity: users rate few movies so few ratings overall
- Biased towards popular movies: many ratings
- Biased against “bad” movies: users don’t rate them
- New movies rated more highly: novelty factor!
- New users rate movies more highly: inexperience?
- Biased towards frequent raters: e.g. teenagers?
- Biased towards English language movies: more users
- Astroturfing: Fake reviews and ratings

Ratings Matrix

- One row for each user; one column for each item
- Each entry r_{ij} is rating of user i for item j
  - Could be explicit rating (stars) or survey response (Likert)
  - Could be 0 (did not watch) or 1 (did watch)
  - Could be derived from how long user interacted with item
- Basic problem is this matrix is very sparse
- Task is to estimate unknown ratings r_{ij} – “matrix completion”
  - ... even when you do not know this yourself

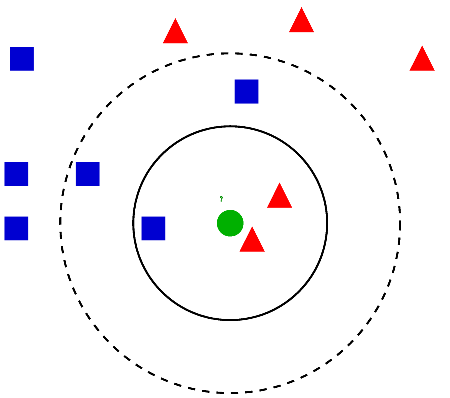
Quiz Question: How many movies are there on IMDb?

Quiz Question: How many has Roger Ebert rated?

User-Based Similarity

- Each user has a row vector containing their ratings (or ‘?’)
  - Notation $\langle r_{u1}, \cdots, r_{un} \rangle$ for n items
- Calculate similarity $sim(u, v)$ between two given users
  - Look only at columns (items) where both entries defined (not ‘?’)
  - Use cosine similarity, Euclidean distance, Pearson correlation, etc.
  - Or Jaccard, Sørensen-Dice, etc., if entries are 0/1
- For user u, find k most similar users $v_1, \cdots, v_k$, for which k?
- Estimate user u’s rating of item i by weighted average of v_j’s ratings
  - $\hat{r}_{ui} = \frac{\sum_{j=1}^{k} sim(u,v_j) r_{v_j i}}{\sum_{j=1}^{k} sim(u,v_j)}$
- Recommend N items to u in order of rating, for which N?

k-Nearest Neighbour Classification



k = 3 ⇒ red triangle; k = 5 ⇒ blue square

User-Based Similarity

Example

							$\cos(u_3,u)$
u_1	7	6	7	4	5	4	0.956
u_2	6	7	?	4	3	4	0.981
u_3	?	3	3	1	1	?	1.0
u_4	1	2	2	3	3	4	0.789
u_5	1	?	1	2	3	3	0.645

Estimate $\hat{r}_{31}$ using 2 most similar users and raw ratings

- $\hat{r}_{31} = \frac{sim(u_3,u_1)r_{11}+sim(u_3,u_2)r_{21}}{sim(u_3,u_1)+sim(u_3,u_2)} = \frac{0.956*7+0.981*6}{0.956+0.981} = 6.48$

User-Based Similarity

Example: Mean-centred ratings

							Mean μ
u_1	1.5	0.5	1.5	−1.5	−0.5	−1.5	5.5
u_2	1.2	2.2	?	−0.8	−1.8	−0.8	4.8
u_3	?	1	1	−1	−1	?	2.0
u_4	−1.5	−0.5	−0.5	0.5	0.5	1.5	2.5
u_5	−1	?	−1	0	1	1	2.0

Estimate $\hat{r}_{31}$ using 2 most similar users to u_3 and mean-centred ratings

- $\hat{r}_{31} = \mu_3 + \frac{sim(u_3,u_1)r_{11}+sim(u_3,u_2)r_{21}}{sim(u_3,u_1)+sim(u_3,u_2)} = 2 + \frac{0.956*1.5+0.981*1.2}{0.956+0.981} = 3.35$

Item-Based Similarity for Ratings

Example: Mean-centred ratings

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	1.5	0.5	1.5	−1.5	−0.5	−1.5
u_2	1.2	2.2	?	−0.8	−1.8	−0.8
u_3	?	1	1	−1	−1	?
u_4	−1.5	−0.5	−0.5	0.5	0.5	1.5
u_5	−1	?	−1	0	1	1
$\cos(i_1,i)$	1.0	0.735	0.912	−0.848	−0.813	−0.990

Estimate $\hat{r}_{31}$ using 2 most similar items to item i_1 and u_3's raw ratings

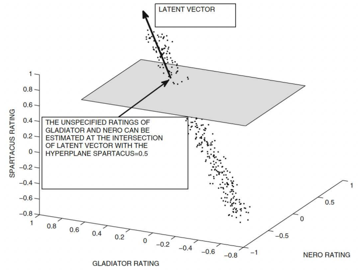
- $\hat{r}_{31} = \frac{sim(i_1,i_2)r_{32}+sim(i_1,i_3)r_{33}}{sim(i_1,i_2)+sim(i_1,i_3)} = \frac{0.735*3+0.912*3}{0.735+0.912} = 3$

User-Based vs Item-Based Similarity

- Item-based recommendations are often more accurate
- User-based recommendations are more diverse
- User-based recommendations include more novel items
- Item-based recommendations can include "explanations"
- Item-based recommendation is more "stable"
- Item-based recommendation is more efficient

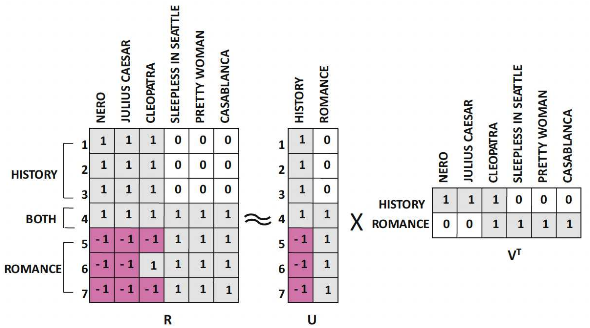
Overall need heuristics with large numbers of users

Matrix Factorization (MF)



- Ratings of (some) movies are highly correlated
- Ratings matrix $m \times n$ is approximately rank $k \ll n$
- Approximate the matrix R using k latent factors
  - $R = UV^T$ where U is $m \times k$, V is $n \times k$

Latent Factors

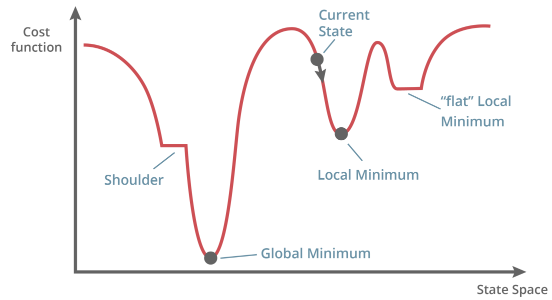


- u_{ik} is how much u's tastes are like k; v_{jk} is how much v_j is like k
- $\hat{r}_{ij} = u_i \cdot v_j$ = sum of $u_{ik}.v_{jk}$ over all the factors k
- Factors are not always interpretable like this!

Unconstrained Matrix Factorization

- Need to find U and V such that error $e = \frac{1}{2}||R - UV^T||^2$ is minimized
  - where $||M|| = \sqrt{\sum_{i,j} m_{ij}^2}$, the Frobenius norm of M
- Here sum over only those values in R that are known
- Stochastic Gradient Descent
  - Start with random matrices for U and V (for which k?)
  - Calculate the error e_{ij} for some r_{ij}, and for some learning rate α, in parallel
    - Update $u_{iq} \leftarrow u_{iq} + \alpha.e_{ij}.v_{jq}$ for each q
    - Update $v_{jq} \leftarrow v_{jq} + \alpha.e_{ij}.u_{iq}$ for each q
  - Until U and V converge

Weight Space



Take small step in the direction that reduces error

Often get stuck in "local minima" or ridges or plateaux

Matrix Factorization with Regularization

- Minimize error $e = \frac{1}{2}||R - UV^T||^2 + \lambda \frac{1}{2}(||U||^2 + ||V||^2)$ for some λ
  - Choose λ using cross-validation or validation set
- Stochastic Gradient Descent
  - Start with random matrices for U and V (for which k?)
  - Calculate the error e_{ij} for some r_{ij}, and for some learning rate α, in parallel
    - Update $u_{iq} \leftarrow u_{iq} + \alpha(e_{ij}.v_{jq} - \lambda.u_{iq})$ for each q
    - Update $v_{jq} \leftarrow v_{jq} + \alpha(e_{ij}.u_{iq} - \lambda.v_{jq})$ for each q
  - Until U and V converge

Alternating Least Squares

- Using either the original or regularized formulation
- Start with random matrices for U and V (for which k?)
  1. Fix matrix U
     - For each row vector $\vec{v}_j$ in V
       - Optimize $\vec{v}_j$ to minimize error $\sum_i (r_{ij} - \vec{u}_i \cdot \vec{v}_j)^2$
  2. Fix matrix V
     - For each row vector $\vec{u}_i$ in U
       - Optimize $\vec{u}_i$ to minimize error $\sum_j (r_{ij} - \vec{u}_i \cdot \vec{v}_j)^2$
- Repeat steps 1 and 2 until U and V converge

Singular Value Decomposition (SVD)

Basic facts from Linear Algebra

- Any matrix A can be decomposed as $U\Sigma V^T$
  - Columns of U and V are mutually orthogonal
  - Σ is diagonal: elements are square roots of eigenvalues of AA^T
  - U and V are unitary: $U^T = U^{-1}$ and $V^T = V^{-1}$
- Define eigenvalue/vector λ, $\vec{v}$ of matrix M: λ, $\vec{v}$ such that $M\vec{v} = \lambda\vec{v}$
  - Columns of U are the eigenvectors of AA^T
  - Columns of V are the eigenvectors of A^TA
  - Eigenvalues λ can be found from the determinant of $AA^T - \lambda I$
  - Eigenvectors for A^TA can be found by solving $A^TA - \lambda I = 0$

This holds if all values in the matrix are known

SVD Example

$$A = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix} \text{ so } AA^T = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 \\ 2 & 3 \\ 2 & -2 \end{bmatrix} = \begin{bmatrix} 17 & 8 \\ 8 & 17 \end{bmatrix}$$

For $AA^T - \lambda I = 0$, $\lambda^2 - 34\lambda + 225 = 0$ so $\lambda = 25, 9, 0$ with singular values 5, 3

$$A^TA - 25 \cdot I = \begin{bmatrix} -12 & 12 & 2 \\ 12 & -12 & -2 \\ 2 & -2 & -17 \end{bmatrix} \approx \begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = 0 \text{ gives unit vector } v_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}$$

Similarly for $\lambda = 3$. Then using $u_i = \frac{1}{\sigma_i} A v_i$ where $\sigma_i = \sqrt{\lambda_i}$ is a singular value

$$AV = U\Sigma \text{ implies } A = U\Sigma V^T = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{18}} & \frac{-1}{\sqrt{18}} & \frac{4}{\sqrt{18}} \\ \frac{2}{3} & \frac{-2}{3} & \frac{1}{3} \end{bmatrix}$$

SVD for Ratings Prediction

- What to do for unknown values?
  - Use mean rating for user, or 0 if ratings are mean-centred?
  - Reorder eigenvalues in order of size $\lambda_1, \cdots, \lambda_k, \cdots$
  - Approximately factorize matrix $R \approx U_k \Sigma_k V_k^T$ where Σ_k is $k \times k$
- Plug in estimated values for unknown ratings and iterate
- Problem is that this introduces bias (and messes with the data ...)

How likely is this to work, given the sparsity of R?

Matrix Factorization Summary

- Some sophisticated algorithms, but what differences in practice?
  - Are recommendations based on MF better than those based on neighbourhoods?
- How good is the data (transactions, playlists, videos watched, etc.)?
  - Especially as users mostly only rate items they like
- How well do these methods scale?
  - How do the methods handle cold start problems (users or items)?
  - Can the matrix be incrementally factorized?

Neighbourhood methods work quite well, though MF can be expected to do well with larger amounts of data – if the above problems can be addressed

Hybrid Recommender Systems

- Weighted ensembles
  - Combine recommendation lists by (learned) weights
- Switching systems
  - Use different recommenders depending on user stage/phase
  - Use recommenders built from different data samples (bagging)
- Cascaded sequences
  - Recommendations are refined by a subsequent model
- Feature augmentation
  - Recommendations are additional features in a subsequent model
- Meta-level: combinations of techniques
  - e.g. content-based profiles used to define neighbourhoods for CF

Weighted Hybrid Methods

Simple approach for ratings prediction

- Take q recommenders that each provide ratings $\hat{r}^i_{uj}$
- Learn weighting α_i for each recommender i
- Combine ratings linearly: $\hat{r}_{uj} = \sum_{i=1}^{q} \alpha_i \hat{r}^i_{uj}$
  - Easy to "learn" such weights using linear regression
  - Use a training/test split to avoid overfitting, with MAE or RMSE
  - Could also try Multi-Layer Perceptron neural network approach
  - Can also try stochastic gradient descent to optimize weights
- Recommenders could be different matrix factorization methods
- Recommenders could be the same method with different values of k

Switching Systems

Might be useful for news recommendation

1. Use content-based recommender, CF and Naive Bayes in sequence
   - Aim for high precision early, then novelty, then long-term interests
2. Have "bucket" of models and choose the best one after retraining
3. Use content-based recommender early in user life cycle, CF later

Not obvious how to blend recommenders/recommendations

Cascaded Systems

- Successive refinement
  - Knowledge-based recommender sorts items into buckets
  - Items ranked within buckets by CF (smaller item set)
- Re-ranking
  - Neighbourhood method gives initial ranking based on score
  - Decision Tree provides content-based "critic"
  - Rules derived from critic adjusts scores down
  - Typically reduces scores of popular items

Feature Augmentation

- Use related-item features from CF in content-based recommender
- Use learned content-based model to define similar users for CF
- Estimate pseudo-ratings with content-based recommender and use CF
- Add derived "ratings" for item features then apply CF
  - Weight the two types of feature differently
- Adjust CF methods to use weighted similarity (boosting)
  - Neighbourhood: Add weights into distances/correlations
  - MF: Add weights to error function to be optimized
- Use meta-features (about the ratings) for each user-item rating
  - Each weight is a learned linear combination of meta-features

Netflix Prize Meta-Features

Whether the user rated more than 3 movies on this date
Log of the number of times the movie has been rated
Log of the number of distinct dates the user has rated movies
Bayesian estimate of mean rating of the movie after subtracting out the user's Bayesian estimated mean
Log of the number of this user's ratings
Standard deviation of this user's ratings
Standard deviation of this movie's ratings
Log of the number of days since this user's first rating
Log of the number of user ratings on this date

Summary

- Collaborative Filtering highly successful
- Often the only choice when content hard to characterize
- Simple neighbourhood methods work well in practice
- Matrix Factorization may work well with large datasets
  - Provided some calculation can be shifted offline
- Other Matrix Factorization methods
  - Non-Negative Matrix Factorization, Probabilistic LSA
- Neural Collaborative Filtering
  - Feed latent vectors into neural network to predict ratings