

# Collaborative Filtering

CS294 Practical Machine Learning

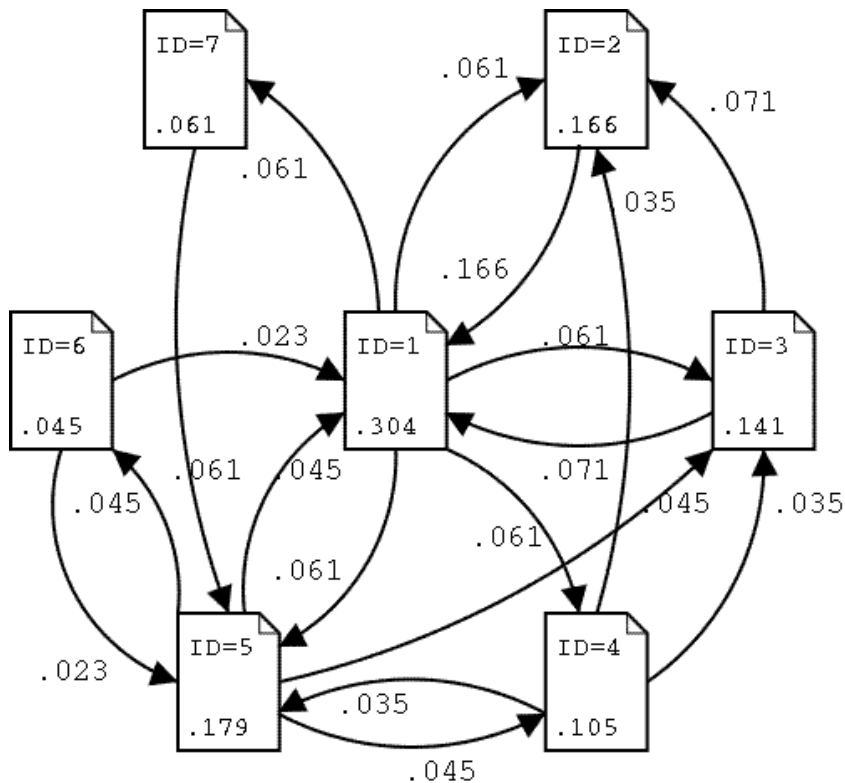
Aleksandr Simma (asimma@cs)

Based on slides by Pat Flaherty





# Google PageRank



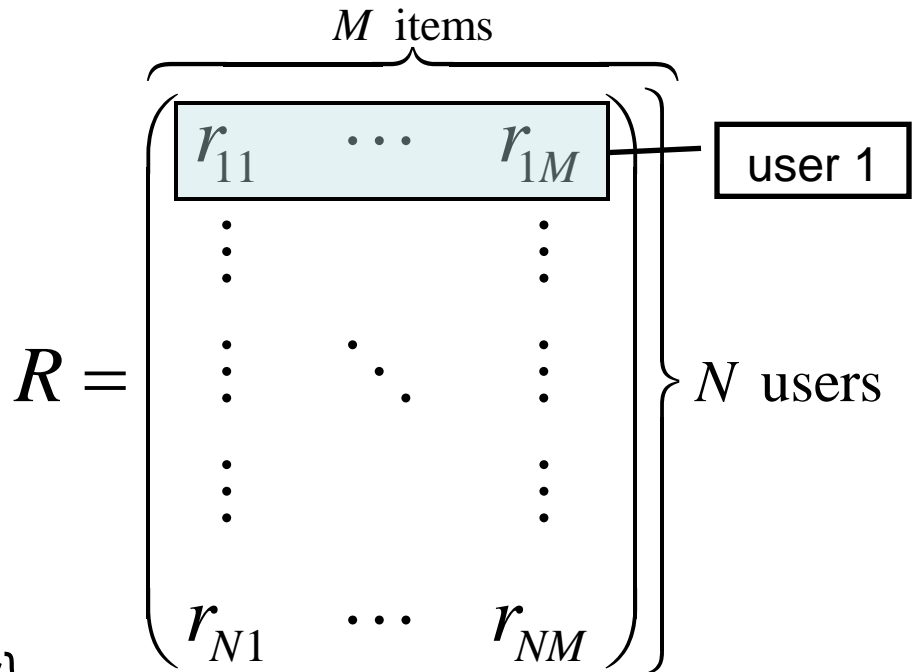
- Collaborative filtering
  - similar users like similar things
- PageRank
  - similar web pages link to one another
  - respond to a query with “relevant” web sites
- Generic PageRank algorithm does not take into account user preference
- extensions & alternatives can use search history and user data to improve recommendations

# Today we'll talk about...

- Problem Formulation
- Classification/Regression
  - Nearest Neighbors
  - Naïve Bayes
- Dimensionality Reduction
  - Singular Value Decomposition
  - Factor Analysis
- Probabilistic Model
  - Mixture of Multinomials
  - Aspect Model
  - Hierarchical Probabilistic Models
- Challenges.

# Input Data

- Users are characterized by their preferences for items (a real number, or an ordinal  $\{1,2,3,4,5\}$ )
- Items are described by the users who prefer them.
- Both can have meta-info.
  - user={age, gender, zip code}
  - item={artist, genre, director}
- Many ratings missing.



sparse rating / co-occurrence matrix

# Problem Formulation

- Usual Setting
  - Want to predict how a user will rate an item.
  - Fill in the missing data in the matrix.
  - Do this based on the observed parts of the matrix (and possibly meta-info)
- Measuring success
  - Mean Average Error
  - Root Mean Square Error
  - Ranking-based objectives.
- Data can be user-entered, or implicitly observed.
  - Ratings v.s. page views, purchase history, etc.

# Two Perspectives

## Usercentric

- Look for users who share the same rating patterns with the query user
- Use the ratings from those like-minded users to calculate a prediction for the query user

## Itemcentric

- Build an item-item matrix determining relationships between pairs of items
- Using the matrix, and the data on the current user, infer his/her taste

**Collaborative filtering:** filter information based on user preference

**Information filtering:** filter information based on content

# Classification and Regression

- Make a classifier or regression to predict a user's rating of an item, as a function of the other ratings the user assigned.
- Each item  $y=1, \dots, M$  gets its own predictor.
  - Plug in a user's observed ratings into a predictor; get a result.
- Reduces collaborative filtering to a well known problem.
- We have very good standard prediction algorithms.
- Need to handle missing data correctly.
- Fit an estimator for each item ... can be a problem
  - Can be a lot of computation
  - Does not use the problem structure.

# Nearest Neighbors

- Compute distance between all other users and query user
  - Only consider items both rated.
- Aggregate ratings from K nearest neighbors to predict query user's rating
  - For real valued ratings, use the mean.
  - For ordinal valued ratings, majority vote or mean.

$r_q$  is the query user

$r_i$  is a training user

$r'_i$  are ratings of items rated by both

$$\begin{aligned}d_{qi} &= d(r_q, r_i) \\ &= \text{cor}(r'_q, r'_i) \\ &= \frac{\text{cov}(r'_q, r'_i)}{\sqrt{\text{var}(r'_q)\text{var}(r'_i)}}\end{aligned}$$

# Weighted kNN Estimation

similarity weights:  $w_{qi} = \frac{1}{d_{qi}}$

$$\hat{r}_q = \arg \max_{r \in D} \sum_{k=1}^N w_{qk} \delta(r, r_k)$$

Majority Vote

$$\hat{r}_q = \frac{\sum_{k=1}^N w_{qk} r_k}{\sum_{k=1}^N w_{qk}}$$

Weighted Mean

- Suppose we want to use information from all users who have rated item  $y$ , not just the nearest  $K$  neighbors
- $w_{qi} \in \{0, 1\} \rightarrow$  kNN
- $w_{qi} \in [0, \infty) \rightarrow$  weighted KNN

# kNN Similarity Metrics

**for** each user

$w_{qu}$  = correlation between query user and each data set user (u)

**end for**

**sort** weights vector

**for** each item

$$\hat{r}_q^y \leftarrow \bar{r}_q + \frac{\sum_{k=1}^K w_{qk} (r_{u_k}^y - \bar{r}_{u_k})}{\sum_{k=1}^K |w_{qk}|}$$

**end for**

- Weights are only computed based on items rated by query and data set users.
- Example: make the weight vector the correlation between q and u
- Possible problem: correlation based on a small number of items may be noisy.

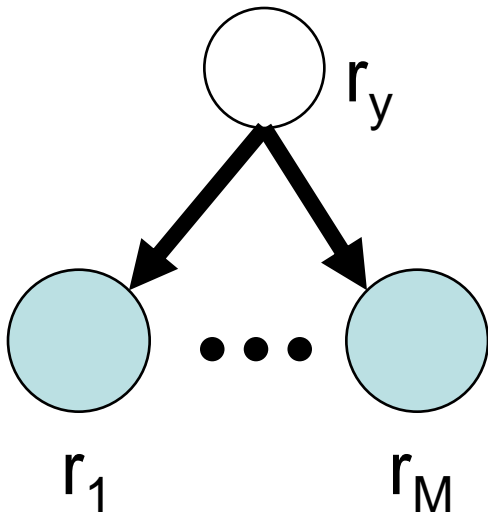
# kNN Complexity

- Must keep the rating profiles for all users in memory at prediction time
- Each item comparison between query user and user  $i$  takes  $O(M)$  time
- Each query user comparison to dataset user takes  $O(N)$  or if using kNN takes  $O(N\log N)$  time to find  $K$
- We need  $O(MN)$  time to compute all rating predictions

# Data set size examples

- MovieLens database
  - 100k dataset = 1682 movies & 943 users
  - 1mn dataset = 3900 movies & 6040 users
- Book crossings dataset
  - after a 4 week long webcrawl
  - 278,858 users & 271,378 books
- Netflix dataset
  - 17700 movies, 250k users, 100 million ratings.
- KDtrees & sparse matrix data structures can be used to improve efficiency

# Naïve Bayes Classifier



- Create one classifier for each item (denote the item  $y$ )
- Main assumption
  - $r_i$  is independent of  $r_j$  given class  $C$ ,  $i \neq j$
  - each user's rating of each item is independent

- prior 
$$\Pr(r^y = v) \propto \sum_{u=1}^N \delta(r_u^y, v)$$

- likelihood

$$\Pr(r^j = w | r^y = v) \propto \sum_{u=1}^N \delta(r_u^j, w) \delta(r_u^y, v)$$

# Classification Summary

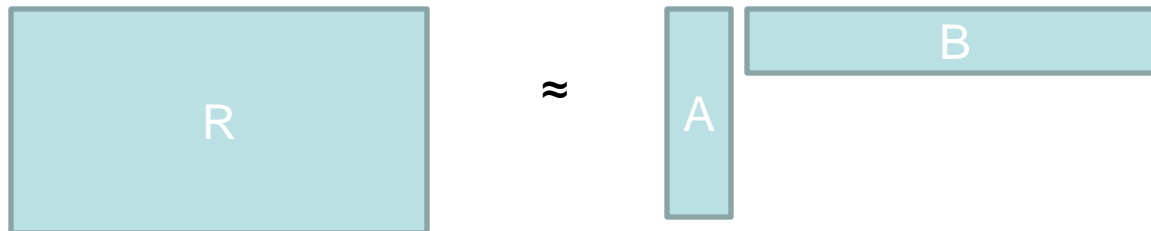
- Any predictor can be used.
  - Predict the rating of an item as a function of other ratings recorded for the user.
  - Must handle missing ratings intelligently.
- Nonparametric methods
  - can be fast with appropriate data structures
  - correlations must be computed at prediction time
  - memory intensive
  - kNN is most popular collaborative filtering method
- Parametric methods
  - Naïve Bayes
  - require less data than nonparametric methods
  - makes more assumptions about the structure of the data

# Today we'll talk about...

- Problem Formulation
- Classification/Regression
  - Nearest Neighbors
  - Naïve Bayes
- Dimensionality Reduction
  - Singular Value Decomposition
  - Factor Analysis
- Probabilistic Model
  - Mixture of Multinomials
  - Aspect Model
  - Hierarchical Probabilistic Models
- Challenges.

# Low Dimensional Matrix Factorization

- We wish to represent the ratings matrix as a product of matrices  $W$  ( $N \times k$ ) and  $V$  ( $k \times M$ )
- Very general idea; many methods can be thought of in this way.
- Consider making rows of  $W$  a probability distribution
  - Each user's ratings is some mixture of distributions, with the mixing weights determined by  $X$
  - Each element of  $V$  could itself be a multinomial distribution for ordinal ratings.



# Singular Value Decomposition

- If all of R is observed, and we use MSE loss, computing is easy.
- Decompose ratings matrix, R, into coefficients matrix A and factors matrix B to minimize

$$\sum_{i=1}^N \sum_{j=1}^M (R_{ij} - [AB]_{ij})^2$$

- This should like PCA to you.
- Use SVD, set  $A=U \Sigma$ ,  $B=V^T$ 
  - U = eigenvectors of  $RR^T$  (N×N)
  - V = eigenvectors of  $R^TR$  (M×M)
  - $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_M)$  eigenvalues of  $RR^T$
  - Take only k top eigenvalues and eigenvectors

# Weighted SVD

$$\arg \min_{A,B} \sum_{i=1}^N \sum_{j=1}^M W_{ij} (R_{ij} - [AB]_{ij})^2$$

- Binary weights
  - $w_{ij} = 1$  means element is observed
  - $w_{ij} = 0$  means element is missing
- Positive weights
  - weights are inversely proportional to noise variance
  - allow for sampling density e.g. elements are actually sample averages from counties or districts

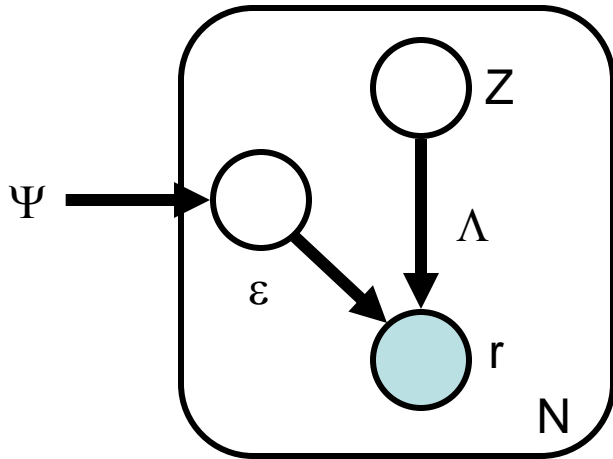
# SVD with Missing Values

- E step fills in missing values of ranking matrix with the low-rank approximation matrix
- M step computes best approximation matrix in Frobenius norm (MSE)
- Local minima exist for weighted SVD
- Can also directly do gradient descent.

*E Step:*  $X = W * R + (1 - W) * \hat{R}$   
where  $*$  is element-wise multiplication

*M Step:*  $[U, \Sigma, V] = SVD(X)$   
 $\hat{R} = U_k \Sigma_k V_k^T$

# PCA & Factor Analysis



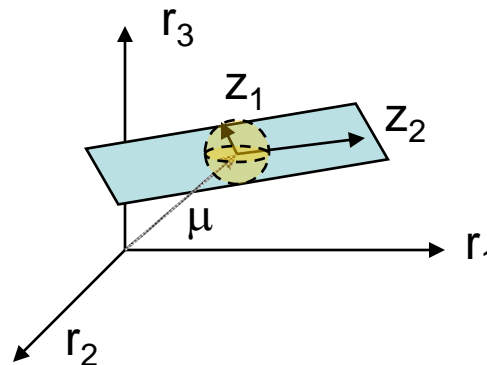
- Factor analysis needs an EM algorithm to work
- EM algorithm can be slow for very large data sets
- Probabilistic PCA:  $\Psi = \sigma^2 \mathbf{I}_M$

- $r$  = observed data vector in  $\mathbb{R}^M$
- $Z$  = latent space  $\mathbb{R}^K$
- $\Lambda$  =  $M \times K$  factor loading matrix
- model:  $r = \Lambda Z + \mu + \varepsilon$
- Integrate out  $z$

$$\varepsilon | \Psi \sim N(0, \Psi)$$

$$z \sim N(0, I_K)$$

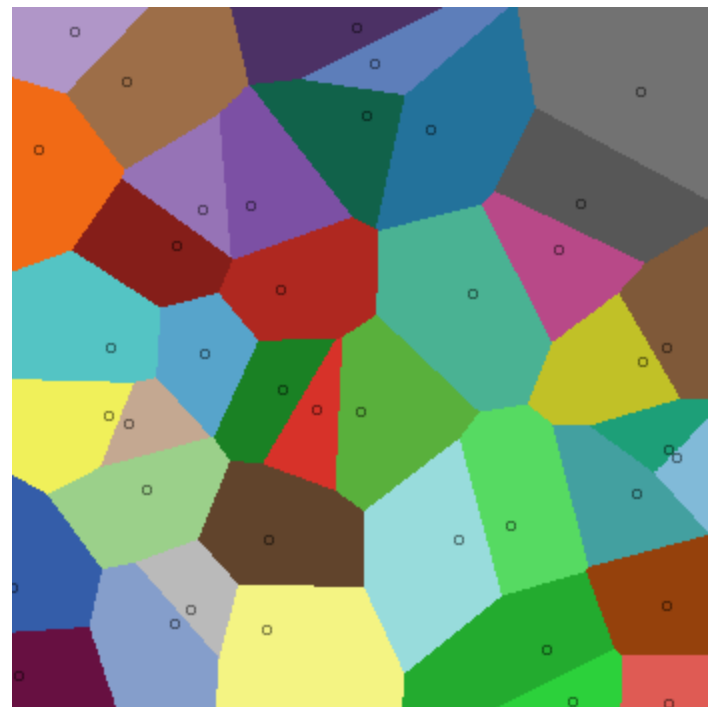
$$r | Z = z; \mu, \Psi, \Lambda \sim N(\mu + \Lambda z, \Psi)$$



$M = 3$   
 $K = 2$

# Matrix methods

- in general: recommendation does not depend on the particular item under consideration – only the similarity between query and dataset users
- one person may be a reliable recommender for another person with respect to a **subset of items**, but not necessarily for all possible items



# Dimensionality Reduction Summary

- Singular Value Decomposition
  - requires one or more eigenvectors (one is fast, more is slow)
  - Weighted SVD
    - handles rating confidence measures
    - handles missing data
  - PageRank
    - only need to solve a maximum eigenvector problem (iteration)
    - “user” preferences incorporated into Markov chain matrix for the web
- Factor Analysis
  - extends Principle components analysis with a rating noise term

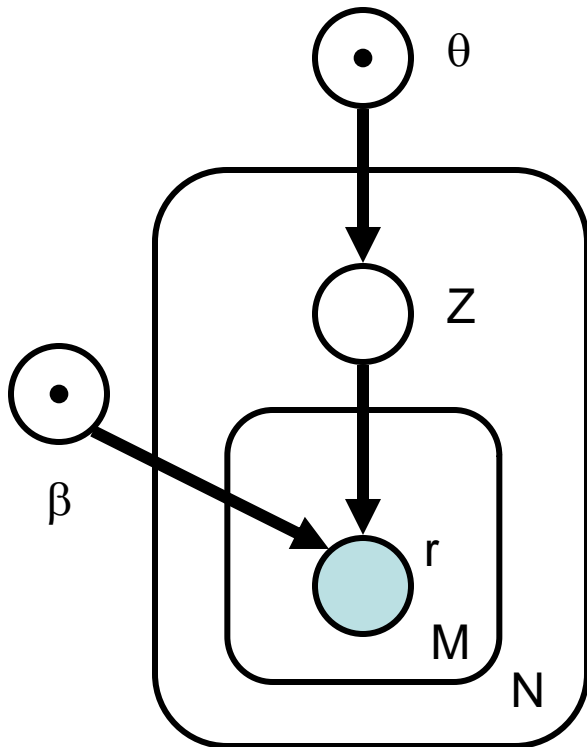
# Today we'll talk about...

- Problem Formulation
- Classification/Regression
  - Nearest Neighbors
  - Naïve Bayes
- Dimensionality Reduction
  - Singular Value Decomposition
  - Factor Analysis
- Probabilistic Model
  - Mixture of Multinomials
  - Aspect Model
  - Hierarchical Probabilistic Models
- Challenges.

# Probabilistic Models

- Nearest Neighbors, SVD, PCA & Factor analysis operate on one matrix of data
- What if we have meta data on users or items?
- Mixture of Multinomials
- Aspect Model
- Hierarchical Models

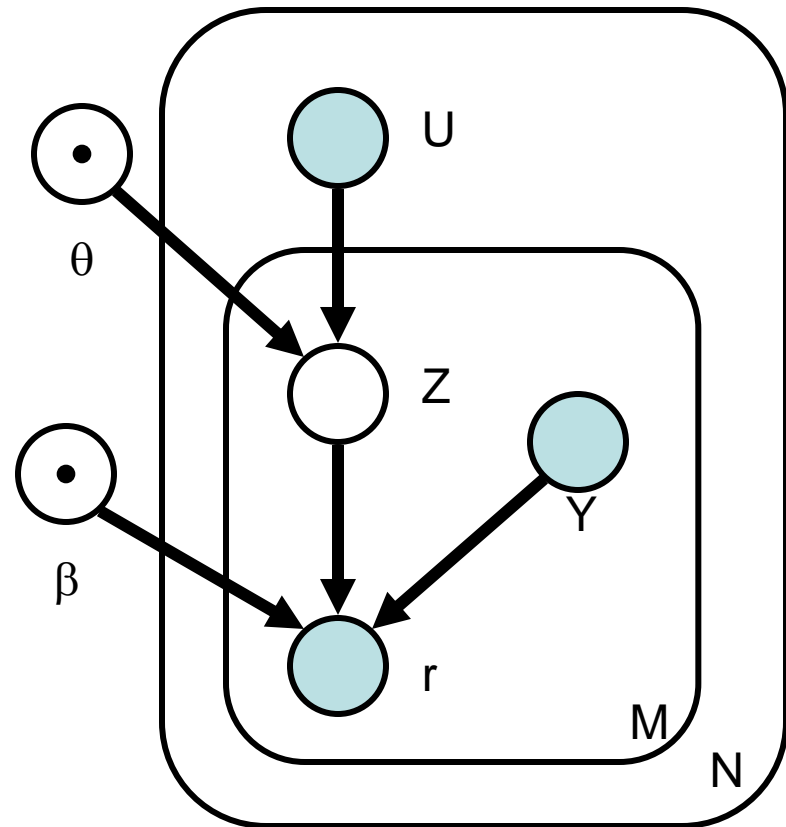
# Mixture of Multinomials



- Each user selects their “type” from  $P(Z|\theta)=\theta$ 
  - Type can also depend on meta-data
- User selects their rating for each item from  $P(r|Z=k) = \beta_k$ , where  $\beta_k$  is the probability the user likes the item.
- User cannot have more than one type.

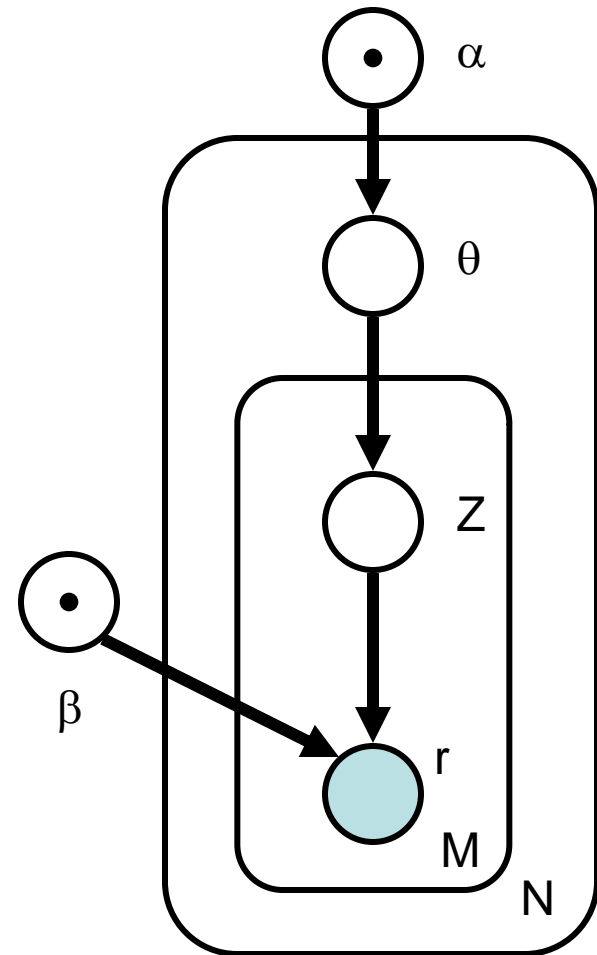
# Aspect Model

- $P(Z|U=u)=\theta_u$
- $P(r|Z=k, Y=y)=\beta_{zk}$
- We have to specify a distribution over types for each user.
- Number of model parameters grows with number of users  
→ Can't do prediction.



# Hierarchical Models

- Meta data can be represented by additional random variables and connected to the model with prior & conditional probability distributions.
- Each user gets a distribution over groups,  $\theta_u$
- For each item, choose a group ( $Z=k$ ), then choose a rating from that distribution  $\Pr(r=v|Z=k)=\beta_k$
- Users can have more than one type (admixture).

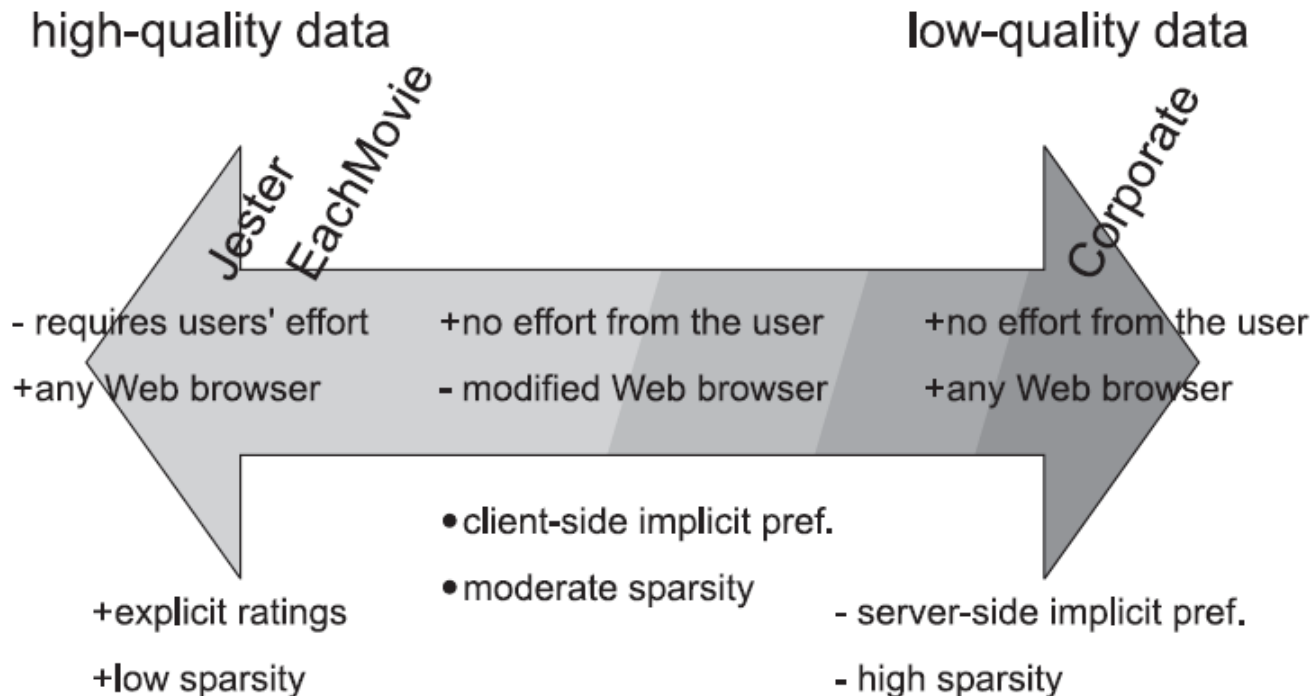


# Cross Validation

- Before deploying a collaborative filtering algorithm we must make sure that it does well on new data
- Many machine learning methods fail at this stage
  - real users change their behavior
  - real data is not as nice as curated data sets

# Support Vector Machine

- SVM is current state of the art classification algorithm
- We conclude that the quality of collaborative filtering recommendations is **highly dependent on the quality of the data**. Furthermore, we can see that kNN is dominant over SVM on the two standard datasets. On the real-life corporate dataset with high level of sparsity, kNN fails as it is unable to form reliable neighborhoods. In this case SVM outperforms kNN.



# Today we'll talk about...

- Problem Formulation
- Classification/Regression
  - Nearest Neighbors
  - Naïve Bayes
- Dimensionality Reduction
  - Singular Value Decomposition
  - Factor Analysis
- Probabilistic Model
  - Mixture of Multinomials
  - Aspect Model
  - Hierarchical Probabilistic Models
- Challenges.

# Challenges

- Good objectives
  - Predicting ratings is not a realistic setting.
  - In reality, may be more interested in rankings, selecting best items, etc.
    - Netflix competition, a lot of effort goes into calibration of user ratings, which is usually irrelevant.
- Missing data.
  - All the methods above assume that the items rated are chosen randomly, independently of preferences.
  - Obviously wrong; how can our models capture information in choices of ratings
    - Marlin et al, UAI07, Salakhutdinov and Mnih, NIPS07

# Challenges

- Other paradigms for collaborative filtering.
  - Active learning?
- How to separate what people like from what people are interested in seeing.
- Multiple individuals using the same account.
- Useful predictions for new users
  - Fixing the missing-at-random assumption helps a lot here.
- Scaling to truly large datasets
  - Simple and parallizable algorithms used in practice.

# Summary

- Non-parametric methods
  - classification + memory-based
    - kNN, weighted kNN
  - dimensionality reduction
    - SVD, weighted SVD
- Parametric Methods
  - classification + not memory-based
    - naïve bayes
  - dimensionality reduction
    - factor analysis
- Probabilistic Models
  - multinomial model
  - aspect model
  - hierarchical models
- Cross-validation

cost: computation & memory  
most popular  
no meta-data

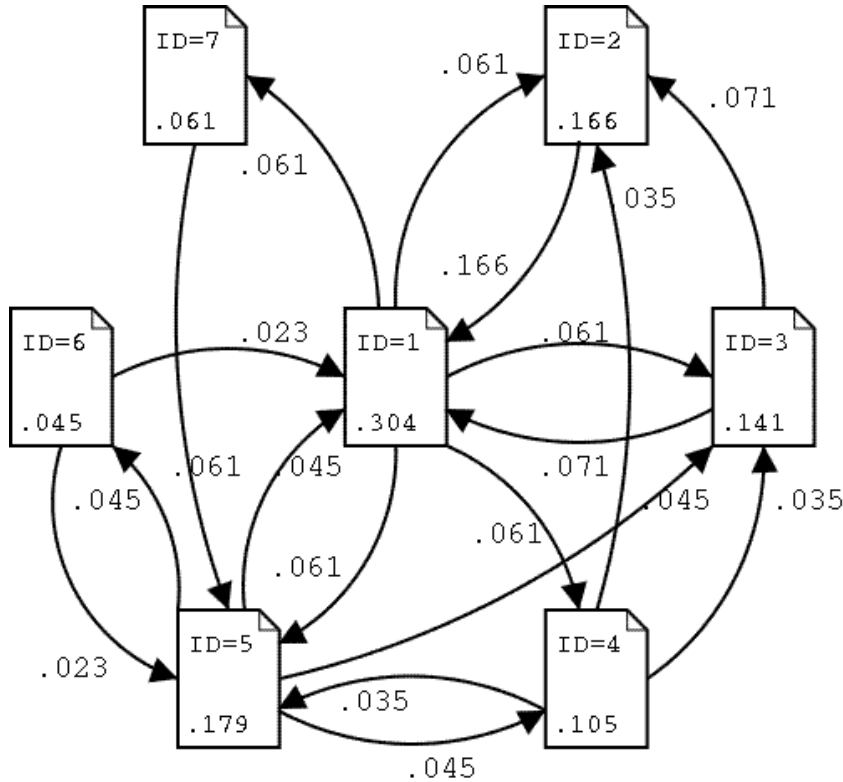
cost: computation  
popular  
missing data ok

cost: computation offline  
many assumptions  
missing data ok

cost: computation offline  
less assumptions  
missing data ok

cost: computation offline  
some assumptions  
missing data ok  
can include meta-data

# PageRank (side note)



word id → web page list

- View the web as a directed graph
- Solve for the eigenvector with  $\lambda=1$  for the link matrix

$$r(P) = \sum_{Q \in B_P} \frac{r(Q)}{|Q|}$$

where  $B_P = \{\text{in degree of } P\}$ ,

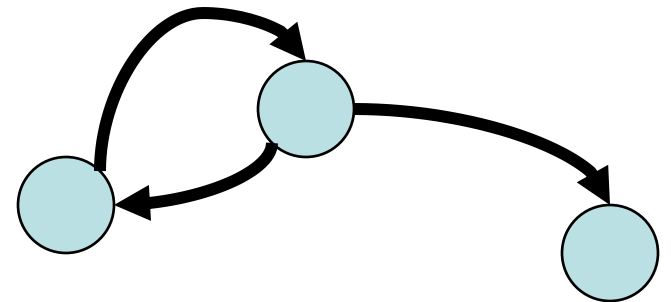
$|Q| = \text{out degree of } Q$

# PageRank Eigenvector

- Initialize  $r(P)$  to  $1/n$  and iterate  $r(P) = \sum_{Q \in B_P} \frac{r(Q)}{|Q|}$
- Or solve an eigenvector problem

$$1\pi = \pi P, \text{ where } p_{ij} = \begin{cases} 1/|P_i| & \text{if } P_i \text{ links to } P_j \\ 0 & \text{otherwise} \end{cases}$$

$$P = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 \end{bmatrix}$$



# Convergence

- If  $P$  is stochastic
  - all rows sum to 1
  - non reducible (can't get stuck), non periodic
- Then
  - the dominant eigenvalue is 1
  - the left eigenvector is the stationary distribution of the Markov chain
- Intuitively, PageRank is the long-run proportion of time spent at that site by a web user randomly clicking links

# User preference

- If the web matrix is not irreducible (a node has no outgoing links) it must be fixed
  - replace each row of all zeros with  $e^T/n$
  - add a perturbation matrix for “jumps”

$$\hat{P} = \alpha P + (1 - \alpha) E, \text{ where } E = \frac{ee^T}{n}$$

- Add a “personalization” vector

$$E = ev^T, \text{ where } v > 0$$

- So with some probability  $\alpha$  users can jump according to a randomly chosen page with distribution  $v$