Recommender Systems

Advanced Machine Learning
Data Mining &
Artificial Intelligence
CSCI E-82
Fall 2018

Peter V. Henstock

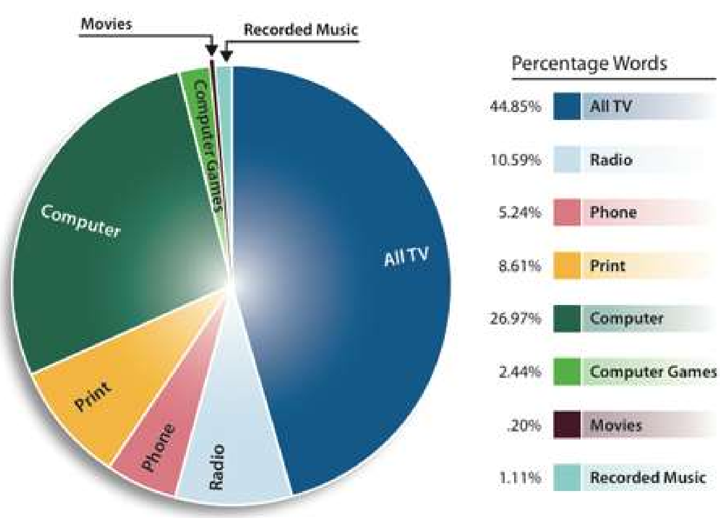
Administrivia

- Lecture: Recommender systems
- Saturday section (Rashmi)
  - Recommender systems
  - Intro to Dask – parallelization
    - Scaling Pandas, scikit-learn, numpy workflows
- Next week lecture December 13:
  - Some reinforcement learning
  - Future of AI and hot topics
  - Review of course
  - Possible final presentations?
- Final project December 20

Administrivia

- Final Presentations
  - Held (some possibly) next week
  - Most held 12/20 on the last day
  - Project due 12/19 – no late days
- Start time 12/20 is TBD
  - Will have survey this week
- Short presentations:
  - Context/problem + what you did + result
  - Request video cameras on

Information: 34 GB/day (2009)



http://bits.blogs.nytimes.com/2009/12/09/the-american-diet-34-gigabytes-a-day/?_r=0

How much data?

- 285 pieces of media content per day
- 54,000 words (63% of network)
- Up to 1000 clickable links
- 443 minutes of video ~ 4 movies

How do we manage to live?

- How can we at least find stuff that is worth our time?
- Ask our friends
  - Seen any good movies lately?
  - Is the iphone XR worth it?
  - Have any good resources for network analysis methods?

Commercial Recommenders

- Amazon
- Yelp
- Match.com/Tinder

- Pretty much every commercial site has some kind of recommender

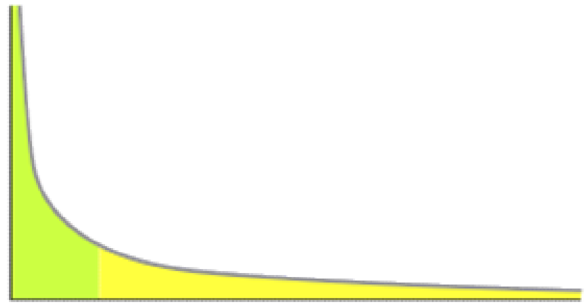
- RecSys
  - Lovestruck.com
  - ArtFinder
  - Big White Wall: mental health advice

Recommender Space

- Brick & mortar has different space
  - Items are limited by space
  - Can direct customers around store
    - Grocery store
  - Can advertise certain items

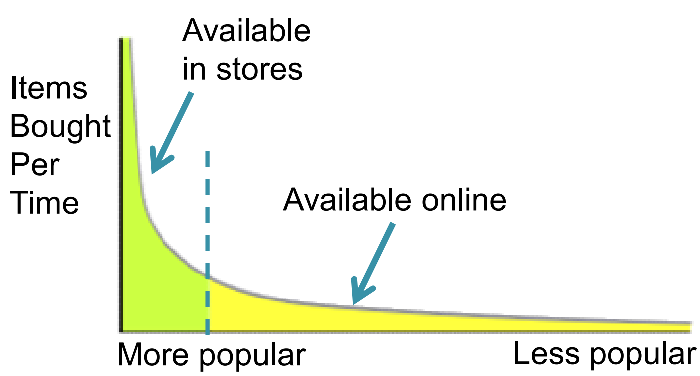
- eStores
  - No limit on shelf space
  - Massive space of items—what to buy?
  - Online market provides too many options
  - Recommender systems

Power Law



- X-axis = rankings of items
- Y-axis = number of items
- Power law describes natural distribution

"Long Tail"



- Power law describes natural distribution
- Many discuss the "long tail" distribution

Search vs. Recommender

- Search (Information Retrieval):
  - One-time query of information
  - Google, etc.
  - No model of a user
- Recommender System
  - Utilizes a user profile
  - Amazon.com
  - Often no query

Recommenders

- Ants leave pheromones as a trail
  - Social navigation
  - Following the crowd
- Information Retrieval
  - Extract relevant information
  - TFIDF
  - Rank the retrieved information by relevance
- Information filtering
  - Filter away useless items
  - Profile creation

Successful Recommenders

amazon

- Reorganized around AI – not one site
- Recommendation engine generates 35% of the company's $180B revenue
  - Forbes July 16, 2018 by Blake Morgan

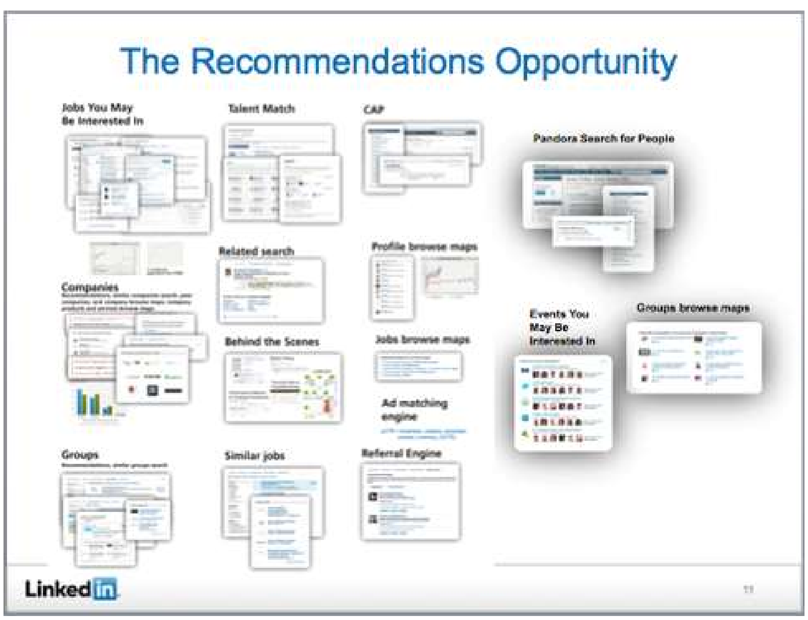
ebay

- Machine Learning is driving incremental sales > $1B/year
  - VentureBeat August 2, 2018

NETFLIX

- Saved $1 Billion from AI
  - Forbes July 9, 2017 by Louis Columbus

LinkedIn: separate recommenders



LinkedIn

- Job Recommendation
- Talent Match when recruiters post
- News recommendation
- Companies you may want to follow
- Groups to join
- PYMK: similar profiles

- 50% of job applications and views are by recommendation

https://www.quora.com/How-does-LinkedIns-recommendation-system-work

History of Recommender Systems

- Manual community filtering
  - Forwarding jokes based on taste
  - Small community has similar tastes
  - 1992 Tapestry system – Xerox Parc

- Collaborative Filtering
  - 1994 GroupLens: U Minnesota
  - 1994 Ringo Music recommendations MIT

- Case Based Reasoning 1980s
  - Roger Schank @yale
  - Initially interest from military

Case-Based Reasoning (CBR)

- Database form used in AI
- Go from specific to general in search
- Know how to solve an ML problem because you've seen something similar before and can apply knowledge
- Adapted to case-based recommendation
- Steer you to personalized approaches
  - Ask questions and have links
  - Easy to explain to the user

Automated Collaborative Filtering

- GroupLens 1994 from Univ. Minnesota
  - Predicted what news articles you would like
  - Nearest neighbor approach based on people preferences
  - Much higher than just average predictions
  - Customized based on similar preferences
- Firefly music recommender from MIT
  - 1995-1999 (RIP) bought by Microsoft in 1998
- Video Recommender from BellCore

GroupLens

- http://grouplens.org/
- Cyclopath (bike routes)
- LensKit for building/researching/studying recommender systems
  - Open-Source
- MovieLens

Recommendation Framework

1) Domain: what recommending?
2) Purpose: sales, help, join
3) Context: how/when it's used
4) Whose opinions:
5) Extent of Personalization:
   - Purchase knowledge of customers
6) Privacy & Personal Information:
7) Output recommendations, scores, etc.
8) Recommendation Algorithm:
   - Content-, Collaborative, etc.

Source: Joseph Konstan

What are recommenders?

- Typically a 2-step process

1) Prediction of a ranking (e.g. 1-5 stars) that a given person would assign to a given item

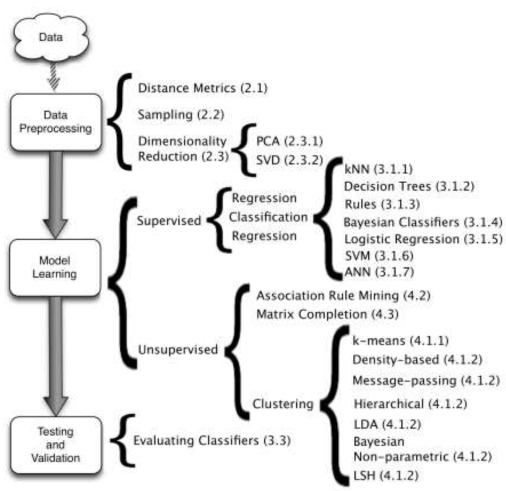
2) Sorting/presentation/interactive process with a user and the ranked values

KDD 2014

Recommendation as data mining

The core of the Recommendation Engine can be assimilated to a general data mining problem

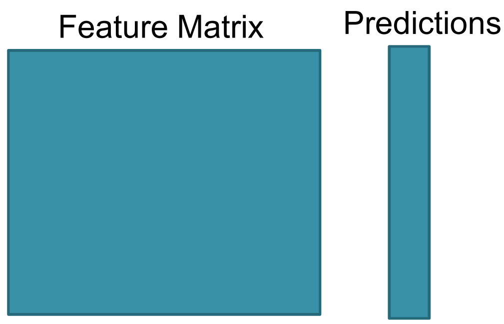
(Amatriain et al. Data Mining Methods for Recommender Systems in Recommender Systems Handbook)



- Xavier Amatriain – August 2014 – KDD

Recommenders & Predictions

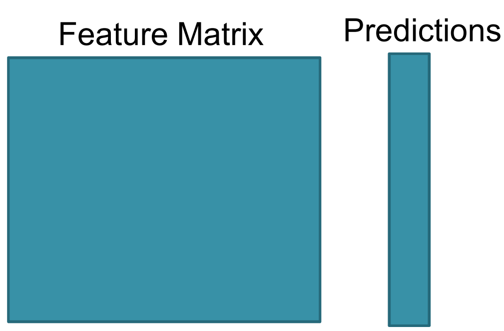
- Isn't a recommender system just the same as prediction?



- How is it different?

Recommenders & Predictions

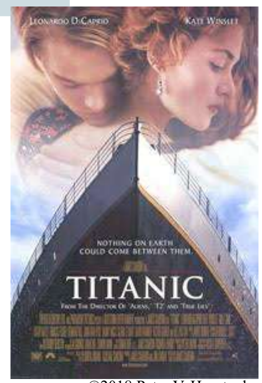
- Isn't a recommender system just the same as prediction?



- What specifically is the Feature Matrix?
- What are the Predictions?

The recommendation problem

	Top Gun	Titanic	Citizen Kane	Toy Story	Brazil
Me		???	1	5	1
Shaggy	5	5	2		1
Scooby	2	2			
Thelma			3		3
Louise	5				



- Assume ratings on 1-5 scale
- Can we easily predict how I would rate Titanic?

Utility Matrix

	Top Gun	Titanic	Citizen Kane	Toy Story	Brazil
Me		???	1	5	1
Shaggy	5	5	2		1
Scooby	2	2			
Thelma			3		3
Louise	5				

- Problem is that the utility matrix is sparse
- Thousands of movies I haven't seen
- Millions of people have seen less
- Main problem: predicting unknowns

The Netflix Problem

Movies

		?	?	?	?	?		?	?	?
	?	?		?		?	?	?	?	?
	?	?	?	?	?	?	?	?		?
	?	?	?		?	?	?	?	?	?
Users		?	?	?	?		?	?	?	
	?		?	?	?	?	?	?		?
	?	?	?	?	?		?	?	?	?
	?	?	?		?	?	?	?		?

- About a million users and 25,000 movies
- Known ratings are sparsely distributed
- Predict unknown ratings

Preferences of users are determined by a small number of factors → low rank

http://0xmachine.com/tutorials/tutorial-icdm2015sparse.pdf

New User on Amazon (or other)

- Just created an account – no info

1) List of top trending Christmas presents
2) Search for L.O.L. Glam Glitter
   - Ranked list of top selling products
   - Non-personalized ranking statistics
3) Might tell system you like x, y, z
   - Builds content-based filtering system for you
4) Might figure out preferences from your purchase/ranking similarity to others
   - Collaborative filtering

Zagat Guide: “Summary Statistics”

- Average a score based on users

Henstock

Non-Personalized Scores

- How should the scoring be done?
- Do you take the average of scores?
- Do you take the median?
- Do you take the percentage of users giving positive recommendations?

Limit of Non-Personalized Scores

- Diversity of consumers
- What kind of restaurant do you like?
  - Caviar and champagne?
  - Greasy spoon diner?
  - Ethiopian fusion?
  - Kid-friendly place?
- Business traveler vs. family vacation?

Product association recommender

- Amazon recommends X goes with Y
  - Iphone case with the iphone
  - Tongs and cover with the grill
- Connections to other categories
- More specific recommendations based on similar purchases
- Context changes from time to product

Frequent Itemsets

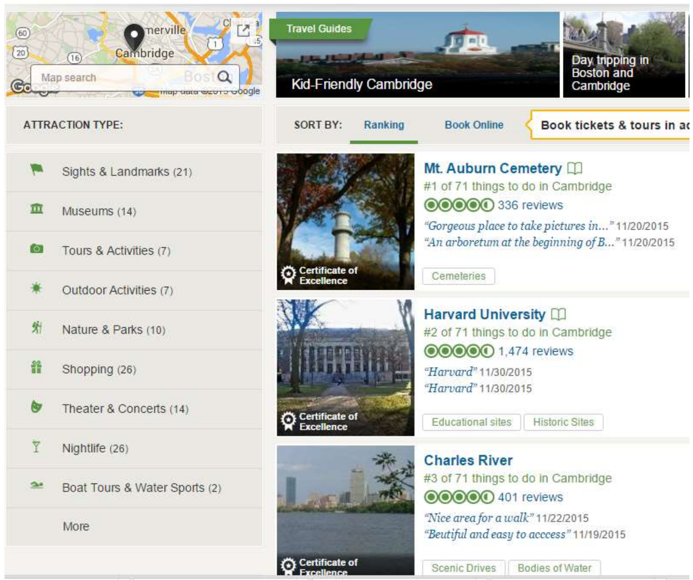
- Examine what items are bought at the same time
- For Home Depot projects
  - Get X, Y and Z
  - Get home and realize I need Y and H
  - Try Y and H and don't have the right tool
  - Next week buy right tool but then use X
- Point = how to define an itemset?

Time-Valued Scores Issue

- Only consider recent scores where hotel has been frequented in past 2 years
- If you went once and hated it
  - You gave it a bad rating
  - You don't return
- If you went many times and liked it
  - You give it a decent rating
  - You return
- Selection bias

TripAdvisor

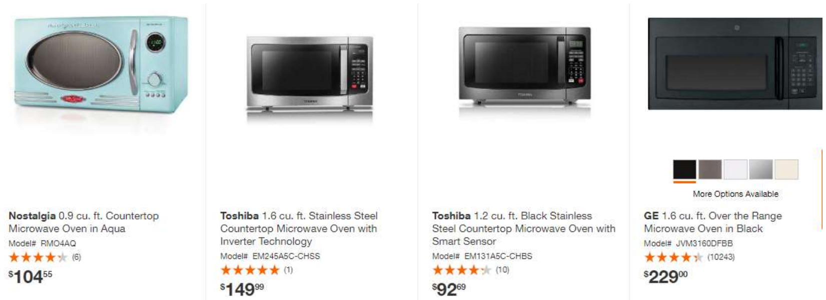
Top Things to Do in Cambridge



How to Present Ranking?

Home Depot microwaves

Best Match 12/6/18



- 4.1 stars 6 reviewers
- 5 stars 1 reviewer
- 4 stars 10 reviewers
- 4.2 stars 10243 reviewers

How do display scores for news?

- Have a bunch of stories and want to display the best stories first
- Users have given + or – for each

- How would you display them?

- www.evanmiller.org/how-not-to-sort-by-average-rating.html

- Score = #positive - #negative
  - Urban dictionary makes this mistake
- Average rating = #positive / total
  - Amazon makes this mistake

Correct Solution

- Lower bound of Wilson score confidence interval for a Bernoulli parameter
- "Given the ratings, 95% chance that the real fraction of positive ratings is at least" this value

$$\left(\hat{p} + \frac{z_{\alpha/2}^2}{2n} \pm z_{\alpha/2}\sqrt{[\hat{p}(1-\hat{p}) + z_{\alpha/2}^2/4n]/n}\right)/(1 + z_{\alpha/2}^2/n).$$

- Only works for +/- rankings

Content-Based Filtering

Personalization w/o Collab Filtering

- Flipboard news feeder
  - Stores interest in machine learning
  - Stores enterprise software engineering
  - Stores scripting languages
  - Stores visualization articles

  - Local restaurants
  - Australian, British, Japanese culture
  - Technology
  - Ballroom dancing
  - Irish fiddling
  - Jazz piano

Personalized Scores

- Previously, it was not tailored for user
- General ranking for everyone

- Don’t really care if the average person likes Taylor Swift

- How would we come up with a system that makes recommendations based on my preferences?

Content-Based Filtering

- Item profiles
  - Set of features describing an item
- User profile:
  - User recommends/scores various items
  - Set of features that user likes
- Main idea:
  - Matching of user profile to item profiles
  - Often use TFIDF for the features since frequent features are kind of boring
  - Idea: weighted sum of features of ranks

Content-Based Filtering

- User ratings
  - User has specific actors they like
  - User has specific genres they like
  - User has news content topics they like
- Generate a content-based profile designed for an end user
- Keyword vector table
  - Update model based on user feedback
  - Use vector table to rate a given movie
  - Use vector dot product

Personalized Recommender

- Use sparse matrix
  - Rows = users
  - Columns = items
  - Cells = ratings
  - How sparse?
- Recommendation:
  - Predict a value for cell in their columns
  - Select a high value for similar user
- Dimensionality reduction on sparse data

Constructing Content Profile

- Ask the user up front but takes time
- Start with generic news and increase customization over time
  - Clicks
  - Time spent on articles
- Might allow user to edit their profile
- Explicit + Implicit

Data Model for Content

- List of attributes that like/hate
- Compare incoming article against list of attributes in list

- Counts of attributes
- TFIDF form of attributes

Content-Based Filtering

- Attribute/Feature based
- Customized news reader like Flipboard

- Do you want to know about the latest news as well?

- How do you balance these?

Challenges of Content-Based

- Need to have well structured data with clear attributes that allow you to be steered toward a given place
- Restaurant:
  - Price, cuisine, parking, wait-time, service
  - Easy to choose restaurant based on these
- Perfect chocolate chip cookie:
  - Kind of soft, chocolate content, sweet, ?
- Good for choosing between alternatives
- Not good for finding complements

Content-Based Filtering

- Advantages
  - Customized for a single user
  - Explainable to the user
  - Can rate a previously unseen item if can obtain features for it
- Disadvantages
  - How to choose features?
    - User could choose features (hard)
    - System could extract features from ratings (hard)
    - Feature space that works is very hard
  - Lack of diversity—no serendipity/surprise
  - “Cold start” problem: new user has no profile

Collaborative Filtering

Content-Based vs. Collaborative

- Content-Based
  - Utilizes attributes or features of objects
  - Prediction based upon attributes
  - Often easy to understand/explain
- Collaborative Filtering
  - Leverages similarities of your tastes against other people's—not just content
  - Prediction based on similar people's preferences
  - Sparse matrix
  - Difficult to explain

Collaborative Filtering

- For music, I like
  - Top 40 alternative
  - 80s rock
  - Irish fiddling
  - Jazz piano
  - J-pop
- Hard to find a coherent set of features that capture these
- Likely people who overlap some of these genres who have ranked their favorites
- Idea: find similar people to me and leverage their rankings to predict mine

Collaborative Filtering

	Top Gun	Titanic	Braveheart	Citizen Kane	Psycho
Peter	5		2	1	
Paul	1		What rank?	3	5
Mary			1		4

- How could we figure out what recommendation we should make for Paul with Braveheart?
- Asking a question of predicting Paul's tastes in movies

Collaborative Filtering

	Top Gun	Titanic	Braveheart	Citizen Kane	Psycho
Peter	5		2	1	
Paul	1		What rank?	3	5
Mary		5	1	3	4

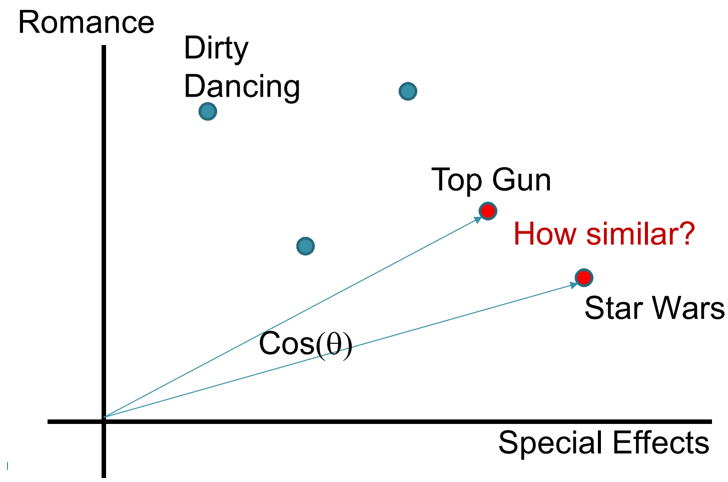
- Strategy:
  - Find people who have similar tastes to Paul
    - Find the closest N users
  - Use their rankings to estimate the rating
    - Take weighted average perhaps

Collaborative Filtering

	Top Gun	Titanic	Braveheart	Citizen Kane	Psycho
Peter	5		2	1	
Paul	1		What rank?	3	5
Mary		5	1		4

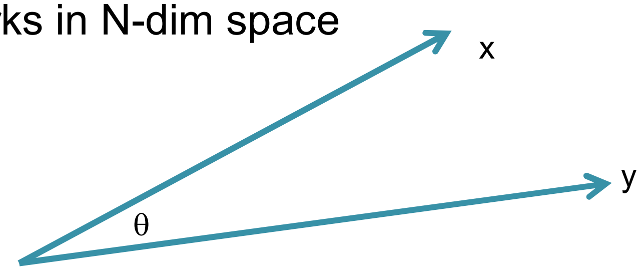
- How to figure out the closest people?
- What distance measure to use?

Vector Concept



Cosine Distance

- $\cos(\theta) = \frac{x \circ y}{||x||||y\}|}$
- Can use the cos(θ) as similarity metric
- Recommend item y to user x when cos(θ) is small
- Works in N-dim space



Collaborative Filtering

	Top Gun	Titanic	Braveheart	Citizen Kane	Psycho
Peter	5		2	1	
Paul	1		What rank?	3	5
Mary		5	1		

- Problem: What to do with missing values?

Pearson Correlation

$$\rho_{X,Y} = \frac{\mathrm{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

- Notice that the values are centered on the mean values in numerator
  - Subtract mean of each row
- Cancels out the effects of having happy or grumpy reviewers
- Allows us to use 0 for missing values
- Cos difference using de-meaned values is equivalent to the Pearson correlation

Collaborative Filtering

- Identify the people who have the most similar tastes
  - Other constraint?

	Top Gun	Titanic	Braveheart	Citizen Kane	Psycho
Peter	5		2	1	
Paul	1		What rank?	3	5
Mary		5	1		4

Collaborative Filtering

- Identify the k people who have the most similar tastes *who have rated the movie*

	Top Gun	Titanic	Braveheart	Citizen Kane	Psycho
Peter	5		2	1	
Paul	1		What rank?	3	5
Mary		5	1		4

- What do we do with them?

Choosing Neighborhood

- Can use all the neighbors
  - Cost issues
  - Not particularly useful
- Take the neighbors within threshold
- Take the closest k nearest neighbors
  - Typical scores are 20-100 depending on the topic and data

Range of scores?

- What if prediction is 5.5 out 1 to 5?
- What if negative similarities?
  - Might exclude neighbors with these

Clustering

- What if you cluster the users?
- Doing blindly doesn't usually work
- Can work if done carefully

Collaborative Filtering

- Identify the people who have the most similar tastes *who have rated the movie*

	Top Gun	Titanic	Braveheart	Citizen Kane	Psycho
Peter	5		2	1	
Paul	1		What rank?	3	5
Mary		5	1		4

- What do we do next?

Collaborative Filtering Equation

- Simple option:
  - $Rating(user, item) = \frac{\sum_{k\ neighbors} Rating(k,item)}{k}$
  - Average of similar users' ratings for that particular item (movie)

- How can we improve upon this?

Collaborative Filtering Equation

- Simple option:
  - $Rating(user, item) = \frac{\sum_{k\ neighbors} Rating(k,item)}{k}$
  - Average of similar users' ratings for that particular item (movie)

- Better option:
  - $Rating(user, item) = \frac{\sum_{k\ neighbors} sim(k,user) * Rating(k,item)}{\sum_{k\ neighbors} sim(k,user)}$

- Other aspect: adjust up/down based on whether user tends to give high/low ratings

Item-Item Collaborative Filtering

- Previously focused on user similarity
- Similar users have similar tastes

- Alternative is to focus on item similarity
- Similar items will have similar properties

- Use exactly same equation replacing k similar users with k similar items
- $Rating(user, item) = \frac{\sum_{k\ neighbors} sim(item,k) * Rating(user,k)}{\sum_{k\ neighbors} sim(item,k\ )}$

Neighborhood of similarity

- How to choose neighborhood?
- Choose the similar items
  - Could choose the k-nearest neighbors
  - Could threshold a distance

- Typically choose k ~ 20

- Precompute item similarities
  - Items are more stable
  - O(#items^2)
  - Tradeoff of k and accuracy

Which works better?

- User-user collaborative filtering
- Item-item collaborative filtering

Which works better?

- User-user collaborative filtering
- Item-item collaborative filtering
- Users are fickle
- Tastes change over time
- Tastes are often not precise
- Represent items with robust features
- Easier to describe items than tastes

Serendipity/Surprise

- User-user collaborative filtering
- Item-item collaborative filtering

- Which method is more likely to let you discovery a new and exciting item through recommendations?

What's the next step?

- *Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion* by Wang et al. 2006
- http://siplab.tudelft.nl/sites/default/files/sigir06_similarityfusion.pdf

- Combines predictions based on ratings of
  - same item by other users
  - similar items by same user
  - similar items by other users

Computational Issues

- Computation to find the neighborhood?

Computational Issues

- Computation to find the neighborhood?
  - For all rows
    - For all columns
      - Compute difference
  - Proportional to #entries in matrix
- How to get around this problem?

Computational Issues

- Computation to find the neighborhood?
  - For all rows
    - For all columns
      - Compute difference
  - Proportional to #entries in matrix
- How to get around this problem?
  - Pre-cluster the users
  - SVD-like dimensionality reduction
  - Pre-compute neighbors

Locally Sensitive Hashing

- Python lshash
  - https://pypi.python.org/pypi/lshash/0.0.4dev
- Hash: x→y
  - Generally a unique key → value mapping
  - Eventually there are "collisions" where duplicates occur
- Relax condition of uniqueness, you can create a set of hashes randomly such that similar entries get mapped together
- Rapid nearest neighbor approach

What goes wrong?

- Harry Potter Effect
  - musicmachinery.com discussion
  - Whimsley.typepad.com
- Collaborative Filtering recommends popular items
- Recommended popular items become more popular
- Feedback loop

What goes wrong?

- Based on assumption that can find people who have similar tastes as you
- What if your tastes shift?
- What is the scope of your taste?
  - If similar movies → similar music? fashion?

What goes wrong?

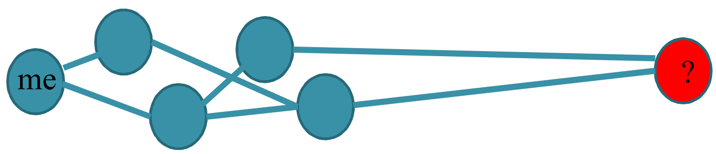
- Yelp has an issue where restaurant owners pay people to recommend the restaurant
- Huge issue for all recommender systems and current area of research
- Detecting fakes
  - Ratings that match others too closely
  - Account creation between times
  - Statistical correlation of distributions

Trust-Based Recommenders

- Previously:
  - Similarity between users → weight
- Now:
  - Trust between users → weight
- Example site: Epinions
- Epinions has/had a "web of trust"
  - Bought by shopping.com in 2003
  - Bought by eBay in 2005

Source & Sink Model

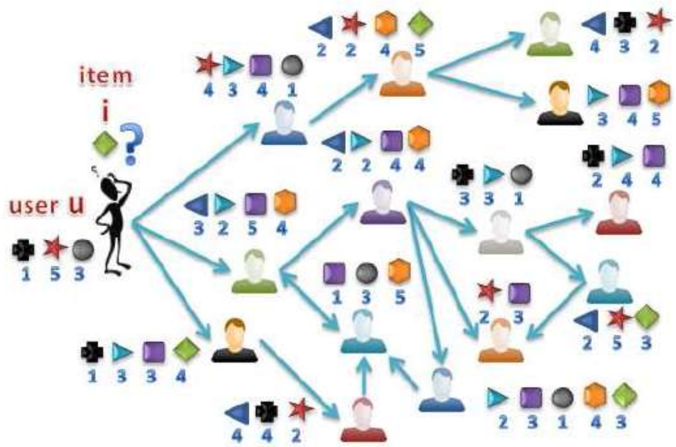
- How to compute trust?
  - Explicit measures for trust:
  - Implicit measures for trust:
    - Different answer depending on who asks
- Source asks friends if they trust sink
- Friends might know directly
- Friends might ask their friends
- Pass back weighted average



Hybrid: Master trusted users

- Some users give great advice and are rewarded with a higher ranking
- High trust makes their comments more valuable
- Notion of global trust
- Recommendation system can weight high trusted users more
- Massa & Avesani 2004

TrustWalker 2009



Advantages & Disadvantages

- Advantages
  - Trust is a somewhat stronger link
  - Generally trust friends and they will give advice for you as compared to generic people with similar tastes
- Disadvantages
  - Hard to create a web of trust

Hybrid Measures

Hybrid Methods

- Similar idea to ensemble approaches
- Balance strengths & weaknesses
- Solving specific problems with your data
  - Might not have enough in some genre
  - Want to leverage other data
  - Aim to improve overall performance
  - “Everything else” category

Feature-Weighted Linear Stacking

- Winner of Netflix prize
- Recommend(u,i) = $\Sigma \alpha_j \text{Recommend}_j(u,i)$
  - Simplest is just weighted sum of different recommenders
- Feature-weighted linear stacking
  - Replace α_j with $f_j(\text{user,item})$
  - Allows generalization to combine different results

Hybridization Methods

Hybridization Method	Description
Weighted	Outputs from several techniques (in the form of scores or votes) are combined with different degrees of importance to offer final recommendations
Switching	Depending on situation, the system changes from one technique to another
Mixed	Recommendations from several techniques are presented at the same time
Feature combination	Features from different recommendation sources are combined as input to a single technique
Cascade	The output from one technique is used as input of another that refines the result
Feature augmentation	The output from one technique is used as input features to another
Meta-level	The model learned by one recommender is used as input to another

https://cow.ceng.metu.edu.tr/Courses/download_courseFile.php?id=2861

Problems/Advantages of Hybrids

- Disadvantages
  - Requires additional tuning
  - Much more difficult to evaluate
  - More computation often involved
- Advantages
  - Widely used in practice
  - Generally try to combine content and collaborative filtering

Evaluation Measures

Evaluation is Difficult

- Data sets: not based on grad students
- Predict all or some of ratings accurately
- What criteria is useful?
  - RMSE?
  - Does business care about RMSE or sales?
- Assume that tastes are constant?
- What about:
  - Diversity
  - Serendipity
  - Non-obvious
  - Recommendations of products already own

Evaluation Methods

- Accuracy/Error Measurements:
- Predictions: AUC, ROC, Precision/Recall
- User action metrics showing influence:
  - Following recommendations
  - How often do users view recommendations
- Business metrics:
  - Sales
  - Lift = increase in busying

Errors

- Predicted vs. Provided recommendation
- 2-stars vs. 2.5-stars

- How can we measure error?

- Mean Absolute Error
  - Average(|Predicted-Actual|
- Root Mean Squared Error
  - Sqrt{Average [Predicted – Actual]^2}

- What are you averaging over?

Netflix Prize

- 1-5 scale
- 100 Million ratings
- 480,000 random customers
- 18,000 movie titles
- 2.8 million customer/movie ID pairs with dates with ratings withheld

- Score used RMSE rounded 0.0001
- Winner 0.8567 (out of 5)
- Is this a good criterion?

Information Retrieval Metrics

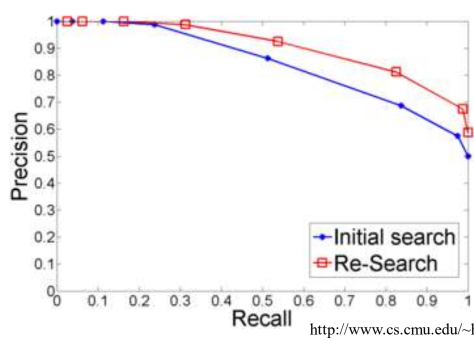
Precision-Recall is similar concept to the ROC curve

Google search returns 10 items

- Precision
  - Percentage returned that are correctly on topic
- Recall
  - Percentage of relevant that are selected

- Plot with a sliding threshold

Precision-Recall & F1



http://www.cs.cmu.edu/~hebert/indexing.html

- $F1 = \frac{2*Precision*Recall}{Precision+Rec}$
- AUC is a single number ~ MAP
  - Mean Average Precision

Recommending for Decisions

- Is there a difference between 2.5 and 2.6?
- May be better to focus on things they will like vs. things they won't
  - Top 10 should contain all things they like
  - Top 10 should not miss things they would like
- "Reversal" is large prediction error that switches the like/hate preference

Recommending for Decisions

- Is there a difference between 2.5 and 2.6?
- "Reversal" is large prediction error that switches the like/hate preference
  - Should you count these instead?

What is the truth for evaluation?

- Recommendations are only relevant to the individual
- Can't hire graduate students to read documents and determine truth
- How do you get the data?
  - Ask lots of questions?
  - Only look at what they recommended?
  - Request immediate feedback?

How to assess rankings?

- Can't show all rankings but maybe only the top 10
  - Assess how good the recommender estimates are against a user's relative preference
- Mean Reciprocal Rank
  - Get a sorted listed of recommendations
  - MRR = 1/j where j=first the user likes
  - Measures #items the users has to view in order to get what they like
    - Broccoli, cauliflower, okra, chocolate
    - ¼ for me since chocolate = #4

Spearman Correlation

- Pearson correlation of the ranks
- Take correlation of ranks to truth

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}.$$

Wikipedia example answer: -0.17575

IQ, X_i	Hours of TV per week, Y_i	rank x_i	rank y_i	d_i	d_i^2
86	0	1	1	0	0
97	20	2	6	−4	16
99	28	3	8	−5	25
100	27	4	7	−3	9
101	50	5	10	−5	25
103	29	6	9	−3	9
106	7	7	3	4	16
110	17	8	5	3	9
112	6	9	2	7	49
113	12	10	4	6	36

With d_i^2 found, add them to find $\sum d_i^2 = 194$. The value of n is 10.

Discounted Cumulative Gain

- In Spearman correlation, all position shifts are equal—but is that useful?
- Notion of half-life
  - User is likely to click the first one item
  - User half as likely to click on 4^{th} if half-life = 3
  - Weight accordingly with decreasing weight
- DCG = $rank1 = \sum_{i=2}^{n} \frac{ranki}{log_2(i)}$
- Normalized DCG = $\frac{DCG}{perfect\ DCG}$

Wikipedia Example: DCG

- DCG = $rank1 + \sum_{i=2}^{n} \frac{ranki}{log_2(i)}$
- Normalized DCG = $\frac{DCG}{perfect\ DCG}$

DCG Table

i	Rank(i)	Log2(i)	Ranki/ log2(i)
1	3	0	N/A
2	2	1	2
3	3	1.585	1.892
4	0	2.0	0
5	1	2.322	0.431
6	2	2.584	0.774
		Total	5.10

Perfect DCG Table

i	Rank(i)	Log2(i)	Ranki/ log2(i)
1	3	0	N/A
2	3	1	3
3	2	1.585	1.262
4	2	2.0	1
5	1	2.322	0.431
6	0	2.584	0
		Total	5.693

- DCG = 3+5.10 perfect DCG=3+5.693
- Normalized DCG = 8/10/8.693 = 0.932

Summary of Ranks

- Normalized DCG
  - most common
- Mean Reciprocal Rank
  - Widely used since it's a no-brainer
- Spearman correlation
  - Liked by statisticians (only)
  - More computational
  - Limited by treating all reorderings the same

Other Considerations

Data Collection

- Preference = what users like
- How can we get this information?
  - Types of reviews
    - #stars – but what does it mean?
    - Vote up/down (Reddit, Pandora, Tivo, Zite)
    - Like
  - When do you review?
    - During song or while reading a link
    - After a movie review/stay
    - A month later

Issues with Rating

- Preferences change over time
- Perception of a song:
  - Annoying when you hear it 10x a day
  - Really great song but don't like it
  - Preference of styles changes
- Rating right after movie vs. a week later may not be the same
- What does a 4.0 mean exactly?

Feedback Types

- Explicit Feedback
  - User ratings
  - Product/restaurant reviews
  - Surveys you've filled in
- Implicit Feedback
  - Clicks
  - Purchases
  - Time on page
  - "follow" (Twitter)
  - "skip" on Pandora
  - Leave in cart

Psychology of Preferences

- Attribute Based
  - Evaluate table based notion
- Experience base: Case Based Reasoning
  - Use experience
- Consequences of actions?
- Sociability: see what other people like

Sequences of Books/Movies

- Police Academy, 2, 3, 4, 5, 6

Harry Potter

Extensions

- Too little data: sparse matrix
  - Collaborative filtering fails
  - Sparse matrix with rare items
  - Few recommenders for item
  - Few items recommended by neighbors
- Opposite: too much data
  - Items do not change
  - Preferences change
  - Nearest neighbor users change
  - Computational challenge

Dimensionality Reduction

Dimensionality Reduction

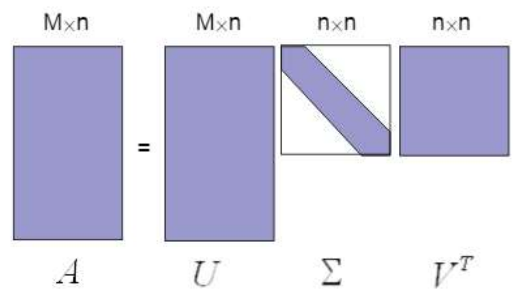
- Rating matrix is sparse
- Columns are generally individual items
  - Not using genres necessarily
  - Not using any ontology typically
- Information Retrieval:
  - Latent Semantic Indexing
  - Mathematical method (PCA/SVD)
  - Maps words into concepts
  - Maps documents into similar clumps

Dimensionality Reduction

- PCA or SVD are dimensionality reduction techniques
- Rank(matrix) = fundamental dimensionality of a matrix
- Sparse matrices will generally have a lower rank and have greater compression
- Likely have a small error when project

SVD for non-square matrices

- Previously we focused on SVD of covariance that was symmetric
- A = Users x Items = U Σ V^T
- U=User features, V=Item-feature
- Reduce right side n→k top (less to store)
- http://slideplayer.com/slide/5186175/



Benefits of Dimensionality Reduction

- ***Identify hidden relationships
  - Similarities of items or users
- **Error of projection can be ignored and attributed to noise
- *Reduce the computation and storage for nearest neighbors, etc.

Utility Matrix Reduction

- Utility matrix = recommender matrix
  - Users = rows
  - Items = columns
- Perform SVD(Utility) → $U\Sigma V^T$
- Choose a subspace of k dimensions
- Dimensions:
- mxn = m x k k x k k x n

SVD Approximation

- Frobenius norm: $||M||_F = \sqrt{\sum_{ij} {M_{ij}}^2}$
- $||Utility-ApproxUtil||_F =$

$$\sqrt{\sum_{ij}(Utility_{ij} - ApproxUtil_{ij})^2}$$

- SVD is optimal in that it minimizes the Frobenius norm for a given k-dimensional approximation

Alternative Notation

- SVD(Utility) → $U\Sigma V^T$
- U are column vectors
- V are column vectors so V^T row vectors
- Σ is a diagonal matrix
- We can do the math are rewrite it as
- SVD(Utility) = $\sum_{i=1}^{k} \sigma_i U_i V_i^T$
- Each U vector ~ user→concept
- Each V vector ~ item→concept
- What is the σ_i on diagonal of Σ?

Other Techniques

- Many approaches for working with sparse matrices such as with collaborative filtering
  - singular value projection (SVP)
  - spectral regularization algorithm (SoftImpute)
  - low rank matrix fitting (LMaFit)
  - alternating minimization (AltMin)
  - boosting type accelerated matrix-norm penalized solver (Boost)
  - Jaggi's fast algorithm for trace norm constraint (JS)
  - greedy efficient component optimization (GECO)
  - Rank-one matrix pursuit (R1MP)
  - Economic rank-one matrix pursuit (ER1MP)

trace norm minimization

alternating optimization

atomic decomposition

- http://0xmachine.com/tutorials/tutorial-icdm2015sparse.pdf

How many dimensions?

- Movies need 13-20 dimensions
- Music needs similar numbers
- Determine #dimensions empirically

- Scaling issue
  - Many weights

Honey, we shrunk the matrix!

- 100 users and 5 movies
- Have a utility matrix 100 x 5

- Perform SVD → $U \Sigma V^T$
- Choose just 2 dimensions so k=2

- Reduced Utility Matrix:
  - U: 100 x 2,
  - V^T: 2 x 5
  - V: 5 x 2

Honey, we shrunk the matrix!

- What do you do with a shrunk matrix?

	Top Gun	Titanic	Braveheart	Citizen Kane	Psycho
Peter	5		2	1	
Paul	1		What rank?	3	5
Mary		5	1		
NewUser	2			3	3

Honey, we shrunk the matrix!

- What do you do with a shrunk matrix?

	Top Gun	Titanic	Braveheart	Citizen Kane	Psycho
Peter	5		2	1	
Paul	1		What rank?	3	5
Mary		5	1		
NewUser	2			3	3

- Need to project a user's preferences into the corresponding "concept" space or latent variable space
- NewUser [2 0 0 3 3] * what? →

Honey, we shrunk the matrix!

- What do you do with a shrunk matrix?

- Need to project a user's preferences into the corresponding "concept" space or latent variable space

	Top Gun	Titanic	Braveheart	Citizen Kane	Psycho
Peter	5		2	1	
Paul	1		What rank?	3	5
Mary		5	1		
NewUser	2			3	3

- NewUser [2 0 0 3 3] * V → 1x2 vector

What have we gained?

What have we gained?

- Less computation
  - Previously we had to math on a much larger matrix for every recommendation
  - Now we can compute an [expensive] reduce utility matrix once every now and then
- Primary advantage is…?

What have we gained?

- Less computation
  - Previously we had to math on a much larger matrix for every recommendation
  - Now we can compute an [expensive] reduce utility matrix once every now and then
- Primary advantage is…?
  - Mapped movies into concepts
  - “Chick-flick romance comedy” concept
  - “Scify with explosions & car chases” concept
  - By mapping, automatically find similarity to concept even if no users actually had the movies in common

Updating the Matrix

- $A = U\Sigma V^T$
- Compute this matrix occasionally (big)

- Can update new data called “fold-in”
- New users can be added
- New items can be added

Problems with SVD

- Computation
  - $O(mn^2)$ where $m \geq n$
  - But fairly fast to predict
  - Needs faster approximations
  - Needs incremental methods for adding data

- Interpretability
  - Linear combination of features

- Sparse Matrix
  - Didn’t deal with this before

Missing Value Problem

- Math assumes full matrix
- Could impute all the missing values
  - Lots of missing values
- Could zero-mean the values
  - Missing values becomes zeros

Convex Optimization

- Goal is to find the user and item features and solve the problem without the full matrix of linear algebra
- A = Users x Items = $U \Sigma V^T$
- Actually not going to compute full A but an approximation to A based on the top k-eigenvalues
- $\min_X ||A-X||_F$: rank(X) = k
  - Frobenius norm
- Achieves the minimum error

Simon Funk blog

http://sifter.org/~simon/journal/20061211.html

Auckland, NZ?

FunkSVD

- Originated in the Netflix prize
- Gradient-descent approach for optimization
  - Uses learning rate and regularization term
  - Minimize predicted error vs. actual
- Train feature one at a time
- Ignore all missing values

- Solve

Incremental SVD Solution

- Start with A matrix
  - U is m x k matrix
  - V is n x k matrix
  - Initialize U and V with small number
- Loop through each of k features
  - Predicted value = dot product u*v for entry
  - $\text{Error}_{ij} = A_{ij} - \text{predicted}_{ij}$
  - Use error to update U and V for entry
    - Take learning rate and regularization into account
    - Uses a gradient descent algorithm

Families of techniques

- SVD++
  - Extension of FunkSVD
- Add in other terms
  - Social networks
  - Recommendations of other media
  - Trust
  - Item similarity
  - Etc.

Optimization

$$\min_{\mathbf{w}\in\mathbb{R}^d} \left\{ f(\mathbf{w}) = l(\mathbf{w}) + r(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} l_i(\mathbf{w}) + r(\mathbf{w}) \right\},$$

Name	Loss function $l_i(\mathbf{w})$
Least Squares	$\frac{1}{2}(y_i - \mathbf{x}_i^T\mathbf{w})^2$
Logistic Regression	$\log(1 + \exp(-y_i\mathbf{x}_i^T\mathbf{w}))$
Squared Hinge Loss	$\max(0, 1 - y_i\mathbf{x}_i^T\mathbf{w})^2$

Name	regularizer (penalty) $r(\mathbf{w})$
Lasso [49]	$\lambda \sum_{j=1}^{d} \lvert w_j \rvert$
Fused Lasso [50]	$\lambda_1 \sum_{j=1}^{d} \lvert w_j \rvert + \lambda_2 \sum_{j=1}^{d-1} \lvert w_j - w_{j+1} \rvert$
Graph Fused Lasso [8]	$\lambda_1 \sum_{j=1}^{d} \lvert w_j \rvert + \lambda_2 \sum_{(j,k)\in\mathcal{E}} \lvert w_j - w_k \rvert$
Group Lasso [65]	$\lambda \sum_{k=1}^{K} \lVert \mathbf{w}_{\mathcal{G}_k} \rVert$
Sparse Group Lasso [13, 44]	$\lambda_1 \sum_{j=1}^{d} \lvert w_j \rvert + \lambda_2 \sum_{k=1}^{K} \lVert \mathbf{w}_{\mathcal{G}_k} \rVert$
Tree Lasso [34, 24]	$\sum_{j=1}^{J} \sum_{k=1}^{K_j} \lambda_k^j \lVert \mathbf{w}_{\mathcal{G}_k^j} \rVert$

http://0xmachine.com/tutorials/tutorial-icdm2015sparse.pdf

Is SVD the best Decomposition?

- Yes, it is optimal but many others
- CUR Decomposition ~ O(mn)
  - Mahoney & Drineas 2008
  - www.pnas.org/content/106/3/697.abstract
  - C = columns from original matrix
  - R = sample of rows from original matrix
  - U = pseudo-inverse of intersection of C&R
- $||A\text{-}CUR||_F <= ||A\text{-}A_K||_F + \varepsilon||A||_F$
  - True with probability of at least 1-delta
- Generally choose 4*k columns and rows

CUR Decomposition

- Different approach
- Typically 4x as many rows and columns
  - Over an order of magnitude bigger
- Not as good a fit

- Why would we do this?
  - Less computation O(mn) vs. O(mn^2)

CUR Row/Col Selection

- Probabilistic selection of columns
  - Define probability of column to be proportional to the length of the column
  - All probabilities have to add to 1 so normalize to sum of all lengths

- Randomly select using probabilities from previous step

- Choose columns and normalize them
- Repeat for rows

U Matrix calculation

- Take the product of C and R
  - C is subset of c columns
  - R is subset of r rows
  - Product will be c x r (matrix multiplication)
  - Called the "intersection" of C and R

- Compute SVD of (c x r)
- "Umatrix" = $U\Sigma V^T$ of intersection except the diagonal of Σ uses $1/\sigma$ instead of σ
- CUR's U matrix is the pseudoinverse

Recent Work

Recommender for Everything

- What time I wake up
- What to eat for breakfast
- Which route to take to work
- Which emails should I read first
- Which courses I should take to maximize my career
- …
- How to connect movie recommenders with music recommenders, etc.

Recent work in Psychology

- How do people make decisions?
- How do people optimize over criteria?
- How do opinions change over time?
- What other factors do users pay attention to?
- How do users view recommendation systems? What do they want?
- Related areas:
  - UI Design with eye-tracking, talk-alouds, etc.
  - A/B Testing
  - Field work: surveys, etc.

Explaining Recommendations

• Transparency:	How system works
• Scrutability:	User says wrong
• Trust:	User confidence in sys
• Effectiveness:	Helps user
• Persuasiveness:	Convince user
• Efficiency:	Faster decisions
• Satisfaction:	Enjoyment/Ease

- Tintarev & Masthoff 2012

Matrix Factorizations

- Tying in additional data on individual
  - Filling in gaps of sparse data
  - Big data era
- Linking social networks to recommendation systems in various ways
- Hybrid work

Influence Limiting

- Prevent undue influence from someone trying to create/bias results by
  - perhaps creating accounts
  - stacking the results
- Influence may be promoting something
- Influence may be attacking competition

Privacy Issues

- Users may be happy contribute but do not want their preferences known
  - Don’t want to be advertising victim
  - Don’t want others to know what they bought
- Encrypted/PGP methods
- Peer-to-Peer recommendations
- Shrinkage methods to compress and anonymize data
- Lots of research in medical data field

Open Source Recommenders

- LensKit (lenskit.org)
- Apache Lucene (lucene.apache.org)
  - SOLR
  - PyLucene
- jCOLIBRI: Case Based Reasoning
  - Gaia.fdi.ucm.es/research/colibri/jcolibri
- Crab (Python) muricoca.github.org
- Recommenderlab for R