Recommendation Systems

CS 5965/6965 - Big Data Systems - Fall 2014

Last time …

- Clustering
- Today
  - Clustering - mapreduce
  - Recommendation Systems
- Assignment
  - Assignment 1 – due tonight
  - Assignment 2
    - Mail TA your ssh public keys → username.key

Clustering using Mapreduce

- User provides k
- Each Map task is assigned a subset of the points
  - Cluster its points
  - Output → $(1, \text{cluster description})$
- Single reducer
  - Merge all clusters

Iterative k-means using mapreduce

- Randomly initialize k centers
- Classify: Assign each point to the nearest center
- Recenter: update centroid
- Repeat until convergence

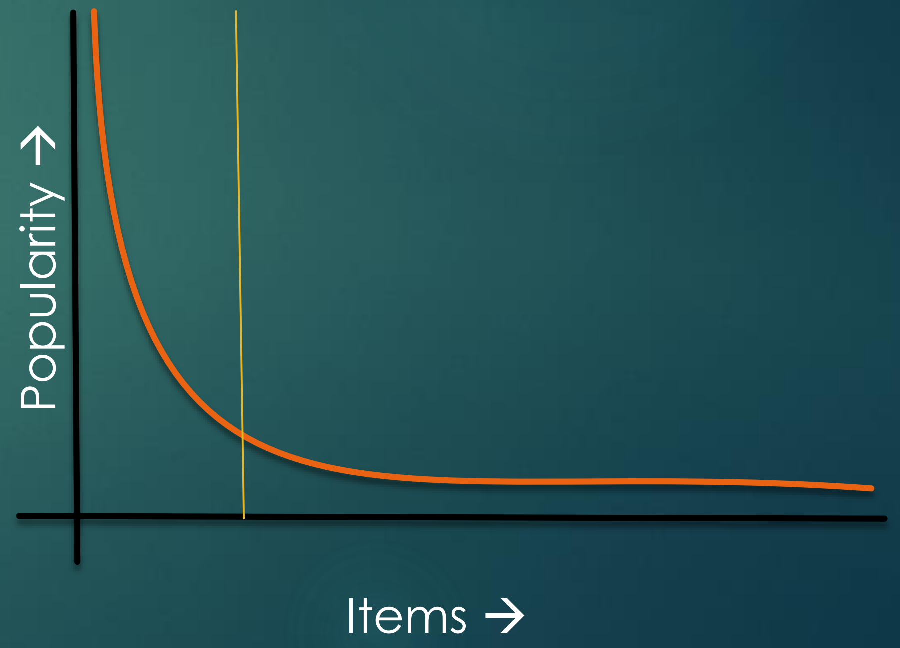
Iterative k-means using mapreduce

- Randomly initialize k centers
- Classify: Assign each point to the nearest center **Map**
- Recenter: update centroid **Reduce**
- Repeat until convergence

Recommendation Systems

Recommendation Systems

- Predicting user responses to options
  - Offering news articles based on users interests
  - Offering suggestions on what the user might like to buy/consume
    - News, Movies, music, books, products
- Physical stores/services
  - Not tailored to individual customer
  - Governed by aggregate numbers
- Online systems
  - Wider variety available → long tail
  - Need to recommend items
    - Ideally tailored to users



Utility Matrix

- Users (rows) & Items (columns)
- Matrix entries are scores/ratings by user for the item
  - Boolean
  - Ordered set
  - Real
- Matrix is sparse

- **Goal of recommendation systems**
  - Predict the blank entries of the utility matrix
  - Not necessary to predict every entry
    - predict some high entries

Populating the Utility Matrix

- Ask users to rate items
  - Not everyone wants to rate
  - Biased – within and across users
- Make inference from users behavior
  - Boolean choice – likes, watched, bought
  - providers like google, amazon have an advantage

- Quality of utility matrix determines the kind of recommendation algorithms that get used

Recommendation Systems

- Two major approaches
  - **Content based systems** – similarity of item properties
    - Depending on the properties of movies you have watched, suggest movies with the same properties – genre, director, actors etc.
  - **Collaborative filtering** – relationship between users and items
    - Find users with a similar ‘taste’
    - Recommend items preferred by similar users

Content-based Recommendations

- Identify user/item profiles and match them for recommendation
- In many cases profiles for items are easy to obtain
  - Movies: genres, actors, director
  - Product description, dimensions, weight
- Harder for others: news articles, blogs

- Example: Search ads
  - Item profiles are categories & keywords for ads
  - User profiles are the keywords user provided for search

Features for documents

- Too much information: a document contains many words
  - A term appearing many times tends to describe the document better
  - Most common words do not carry any significance
    - ‘the’, ‘and’, … → stop words
- TD.IDF (Term frequency . Inverse document frequency)
  - TD → how frequent in a particular term in a document
  - IDF → how frequent is a particular term across documents
- Similarity
  - Jaccard distance → intersection size / union size
  - Cosine distance → normalized inner product

Obtaining User profiles

- Probably the most valuable data are those that contain user activities or behavior
- Direct: search keywords, filing out profiles/surveys
- Indirect:
  - Blogposts, tweets
  - Browsing history

Making recommendations

- Similarity between users and items profiles
  - Jaccard, cosine, any other metric
- Use some bucketing technique to find items
  - Trees, Hashing
- Classification algorithms
  - Using users ratings, learn users ‘taste’
  - Predict ratings for other items

Collaborative Filtering

- Instead of using an item-profile vector use the column in the utility matrix
  - Item defined by which users have bought/rated the item
- Instead of using an user-profile vector use the row in the utility matrix
  - User defined by what items they have bought/liked

- Users similar if their vectors are close using some metric
  - Jaccard, cosine
- Recommendations based on finding similar users and recommending items liked by similar users

Measuring similarity

- Sparsity of utility matrix poses some challenges

- Rounding data
  - Consider 3,4,5 as 1 and 1,2 as 0 → same as unwatched
  - Jaccard distance
- Normalizing ratings
  - Subtract average user rating from each rating
  - Convert low ratings into negative numbers
  - Cosine distance

Duality of Similarity

- Two approaches estimate missing entries of the utility matrix
  - Find **similar users** and average their ratings for the particular item
  - Find **similar items** and average user's ratings for those items

- Considerations
  - Similar users: only find similar users once, generate rankings on demand
  - Similar items: need to find similar items for all items
    - Is more reliable in general

Clustering users and items

- In order to deal with the sparsity of the utility matrix
- Cluster items
  - New utility matrix has entries with average rating that the user gave to items in the cluster
  - Use this utility matrix to …
- Cluster users
  - Matrix entry → average rating that the users gave
- Recurse
  - Until matrix is sufficiently dense

Estimating entries in the original utility matrix

- Find to with clusters the user (U) and item (I) belong, say C and D
- If an entry exists for row C and column D, use that for the UI entry of the original matrix
- If the CD entry is blank, then find similar item (clusters) and estimate the value for the CD entry and consequently that for the UI entry of the original matrix.

Dimensionality reduction

- Utility Matrix, M, is low rank → Singular Value Decomposition

- $M \rightarrow n \times m$
- $M = UV$
- $U \rightarrow n \times d, \quad V \rightarrow d \times m$

- How close is UV to M → root mean square error (RMSE)
  - Sqrt of Sum of difference over all nonblank entries

Incremental computation of UV

- Preprocess matrix M
- Start with an initial guess for U, V
- Iteratively update U, V to minimize the RMSE
  - Optimization problem

Preprocessing M

- Normalize for user
  - Subtract average user rating
- Normalize for item
  - Subtract average item rating
- Both
  - Subtract average of user and item rating from m_{ij}

- Need to undo normalization while making predictions ...

Initializing U, V

- Need a good guess
- Some randomness helps
- Initialize all entries to the same value
  - 0 is a good choice if normalized
  - Else, $\sqrt{\frac{a}{d}}$ is a good value
- Ideally start with multiple initial guesses centered around 0

Optimizing

- Gradient descent
- First order approximation
- Update using steps proportional to the negative gradient of the objective function (RMSE)
- Stop when gradient is zero
- Inefficient for large matrices
  - Stochastic Gradient descent
  - Randomized SVD

Next lecture