CS276B
Text Information Retrieval, Mining, and Exploitation

Lecture 10
Feb 18, 2003

Recap – last time
- Vector space classification
- Nearest neighbor classification
- Support vector machines
- Hypertext classification

Today’s topics
- Recommendation systems
- What they are and what they do
- A couple of algorithms
- Going beyond simple behavior: context
- How do you measure them?
  - Begin: how do you design them "optimally"?

Recommendation Systems
- Given a set of users and items
  - items could be documents, products, other users ...
- Recommend items to a user based on
  - past behavior of this and other users
  - additional information on users/items.

Sample Applications
- Corporate Intranets
  - Recommendation, finding domain experts, ...
- Ecommerce
  - Product recommendations - amazon
- Medical Applications
  - Matching patients to doctors, clinical trials, ...
- Customer Relationship Management
  - Matching customer problems to internal experts in a Support organization.

Corporate intranets - document recommendation
Inputs to system

- Behavior
  - users’ historical “transactions”
- Context
  - what the user appears to be doing now
- Role/domain
  - additional info about users, documents ...

Inputs - more detail

Past transactions from users:
- which docs viewed
- content/attributes of documents
- which products purchased
- pages bookmarked
- explicit ratings (movies, books ... )

Current context:
- browsing history
- searches issued

Explicit role/domain info:
- Role in an enterprise
- Document taxonomies
- Interest profiles

Example - behavior only

<table>
<thead>
<tr>
<th>Users</th>
<th>Docs viewed</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>d1, d2, d3.</td>
</tr>
<tr>
<td>U2</td>
<td>d1, d2.</td>
</tr>
</tbody>
</table>

Recommend d3 to U2.

Expert finding - simple example

![Diagram showing relationship between U1, U2, and d1, d2, d3.]

Recommend U1 to U2 as someone to talk to?

Simplest Algorithm

![Diagram showing relationship between U and d1, d2, d5.]

U viewed d1, d2, d5.
Look at who else viewed d1, d2 or d5.

Recommend to U the doc(s) most “popular” among these users.
Simple algorithm - shortcoming
- Treats all other users as equally important
- Ignores the fact that some users behaved more like me in the past

Measuring collaborative filtering
- How good are the predictions?
- How much of previous opinion do we need?
- Computation.
- How do we motivate people to offer their opinions?

Other aspects
- Rule-based recommendation
- Working in user space vs. item space
- Build regression models of user

Rule-based recommendations
- In practice – rule-based systems in commerce engines
  - Merchandizing interfaces allow product managers to promote items
  - Criteria include inventory, margins, etc.
- Must reconcile these with algorithmic recommendations

User space vs. item space
- Should we work with user similarity or item similarity?
- As with general clustering
  - Recommendations could come from similar users
  - Or opinions could come from items similar to the one we seek an opinion on
  - Similar based on what?
- In some cases, can use both cues

Matrix view
\[ A = \begin{bmatrix} A_{ij} \end{bmatrix} \]
\[ A_{ij} = \begin{cases} 1 & \text{if user } i \text{ viewed doc } j, \\ 0 & \text{otherwise.} \end{cases} \]
AA': Entries give # of docs commonly viewed by pairs of users.
Voting Algorithm

- Row i of AA^T: Vector whose j^th entry is the # of docs viewed by both i and j.
- Call this row r_j, e.g., (0, 7, 1, 13, 0, 2, ....)

What’s on the diagonal of AA^T?

Voting Algorithm - implementation issues

- Wouldn’t implement using matrix operations
  - use weight-propagation on compressed adjacency lists
- Need to log and maintain “user views doc” relationship.
  - typically, log into database
  - update vote-propagating structures periodically.
- For efficiency, discard all but the heaviest weights in each r_j
  - only in fast structures, not in back-end database.

Write pseudo code

Exercise

- The voting algorithm may be viewed as one iteration of the Hubs/Authorities algorithm from CS276a (as in Lecture 3).
- Derive the extension to the full Hubs/Authorities algorithm with convergence.
- Make sure all users don’t get the same recommendations!
- How do you interpret the top “Hubs”?

Different setting/algorithm

- Each user i rates some docs (products, ...)
  - say a real-valued rating U_{ik} for doc k
  - in practice, one of several ratings on a form
- Thus we have a ratings vector U_i for each user
  - (with lots of zeros)
- Compute a correlation coefficient between every pair of users i,j
  - dot product of their ratings vectors
  - (symmetric, scalar) measure of how much user pair i,j agrees: S_{ij}

Predict user i’s utility for doc k

- Sum (over users j such that U_{jk} is non-zero) S_{ij} U_{jk}
- Output this as the predicted utility for user i on doc k.

So how does this differ from the voting algorithm?

I really doesn’t...
Same algorithm, different scenario
- Implicit (user views doc) vs. Explicit (user assigns rating to doc)
- Boolean vs. real-valued utility
  - In practice, must convert user ratings on a form (say on a scale of 1-5) to real-valued utilities
  - Can be fairly complicated mapping
    - Likeminds function (Greening white paper)
  - Requires understanding user’s interpretation of form

Rating interface

Early systems
- GroupLens (U of Minn) (Resnick/lacovou/Bergstrom/Riedl)
- netPerceptions company
- Tapestry (Goldberg/Nichols/Oki/Terry)
- Ringo (MIT Media Lab) (Shardanand/Maes)
- Experiment with variants of these algorithms

Recap slide 6 - Inputs
Past transactions from users:
- which docs viewed
- content/attributes of documents
- which products purchased
- pages bookmarked
- explicit ratings (movies, books ...)

Current context:
- browsing history
- searches issued

Explicit profile info:
- Role in an enterprise
- Document taxonomies
- Interest profiles

The next level - modeling context
- Suppose we could view users and docs in a common vector space of terms
  - docs already live in term space
- How do we cast users into this space?
  - Combination of docs they liked/viewed
  - Terms they used in their writings
  - Terms from their home pages, resumes ...

Context modification
- Then "user u viewing document d" can be modeled as a vector in this space: u * d
- User u issuing search terms s can be similarly modeled:
  - add search term vector to the user vector
- More generally, any term vector (say recent search/browse history) can offset the user vector
Using a vector space

- Similarities in the vector space used to derive correlation coefficients between user context and other users

Recommendations from context

- Use these correlation coefficients to compute recommendations as before
- Challenge:
  - Must compute correlations at run time
  - How can we make this efficient?
  - Restrict each user to a sparse vector
  - Precompute correlations to search terms
  - Compose \( u + cs \)

Correlations at run time

- Other speedup
  - If we could restrict to users “near” the context
  - Problem - determining (say) all users within a certain “ball” of the context
  - Or \( k \) nearest neighbors, etc.

Modified vectors

- Should context changes to vector be made permanent?
- Exponential decay?
- Can retain some memory of recent search/browse history

Think of how to do this efficiently.

Measuring recommendations

- Typically, machine learning methodology
- Get a dataset of opinions; mask “half” the opinions
- Train system with the other half, then validate on masked opinions
  - Studies with varying fractions ≠ half
- Compare various algorithms (correlation metrics)

\[ k \text{ nearest neighbors - efficacy} \]

Source: Sarwar/Karypis/Konstan/Riedl
Summary so far

- Content/context expressible in term space
- Combined into inter-user correlation
  - This is an algebraic formulation, but
  - Can also recast in the language of probability
- What if certain correlations are “constrained”
  - two users in the same department/zip code
  - two products by the same manufacturer?

Recap slide 6 - Inputs

Past transactions from users:
- which docs viewed
- content/attributes of documents
- which products purchased
- pages bookmarked
- explicit ratings (movies, books … )

Current context:
- browsing history
- searches issued

Explicit profile info:
- Role in an enterprise
- Document taxonomies
- Interest profiles

Capturing role/domain

- Additional axes in vector space
  - Corporate org chart - departments
  - Product manufacturers/categories
  - Make these axes “heavy” (weighting)
- Challenge: modeling hierarchies
  - Org chart, product taxonomy

Measuring recommendations

- Unclear how to design correlation metric to yield good results
- How can we tune the algorithm “up front”??
- Need a formulation of what the system is trying to do

Utility formulation

- Microeconomic view again
- Assume that each user has a real-valued utility for each item
- $m \times n$ matrix of utilities for each of $m$ users for each of $n$ items
  - not all utilities known in advance
  - (which ones do we know?)
- Predict which (unseen) utilities are highest for each user

User types

- If users are arbitrary, all bets are off
  - typically, assume matrix is of low rank
  - say, a constant $k$ independent of $m,n$
  - some perturbation is allowable
- I.e., users belong to $k$ well-separated types
  - (almost)
  - Most users’ utility vectors are close to one of $k$ well-separated vectors
Matrix reconstruction

- Given some utilities from the matrix
- Reconstruct missing entries
  - Suffices to predict biggest missing entries for each user
  - Suffices to predict close to biggest
  - For most users
- This is the formulation we will begin with next time

Resources

- GroupLens
  - http://citeseer.nj.nec.com/resnick94grouplens.html
- Shardenand/Maes
  - http://citeseer.nj.nec.com/shardenand95social.html
- Sarwar et al.
  - http://citeseer.nj.nec.com/sarwar01itembased.html