RECOMMENDER SYSTEMS

MOHAMED ELSAIED

02/06/2019
AGENDA

- Motivation
- Problem Formulation
- Types of Recommender Systems
- Content-based Filtering
- Collaborative-based Filtering
- Case Study
RECOMMENDATIONS

Search

Recommendation

ITEMS

Products, web sites, blogs, new items, …

http://dbis.informatik.uni-freiburg.de/lehre/WS1516/Projekt/Various+Aspects+of+Recommender+Systems/
SCARCITY TO ABUNDANCE

• Physical places does have shelf space which has a real estate cost, so limited number of items can be placed
  • Also TV, theaters, etc …

• The web has no shelf space limitations
  • From scarcity to abundance
  • “Long Tail” phenomenon arises
THE LONG TAIL

Area under the curve under the long tail is as big or even bigger than the left of the cut off
Editorial & Hand Crafted
- List of favorites
- List of essential items
- Not scalable
- Not personalized

Problem with new users, items
- Popularity Bias
MATHEMATICAL MODEL

- \( C = \text{set of Customers} \)
- \( S = \text{set of Items} \)
- Utility Function \( u: C \times I \rightarrow R \)
- \( R = \text{set of Ratings} \)
- Likert Scale
- Ordinal data
## Utility Matrix

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
</tr>
</thead>
<tbody>
<tr>
<td>James</td>
<td>1</td>
<td>?</td>
<td>0.2</td>
<td>?</td>
</tr>
<tr>
<td>Megan</td>
<td>0.5</td>
<td>0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Michael</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>Vincent</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td></td>
</tr>
</tbody>
</table>
KEY PROBLEMS

• Gathering Known ratings for matrix
  • How to collect the data in the utility matrix

• Predict unknown ratings from the known ones
  • Mainly interested in high unknown ratings

• Evaluation of recommender systems
  • Evaluation metrics and measure of performance
GATHERING RATINGS

• Explicit
  • People rating items
  • Not scalable
  • Simple & Direct

• Implicit
  • Learn ratings from users
  • Sentiment Analysis
  • Twitter
PREDICTING UNKNOWN RATINGS

- Challenge
  - Most items are not rated
  - Utility matrix is sparse
  - Cold start
    - New items have no ratings
    - New users have no history
CONTENT BASED

- Recommend items by customer \( x \) rated similar to previous items rated highly by \( x \)
- Uses implicit & explicit ratings
- For each item, create item profile
- Profile is a set of features (vector)
FEATURES EXAMPLES

• Text features … find set of important words

• Term Frequency Inverse document frequency (TF-IDF)
  • If a word appears frequently in a document, it's important. Give the word a high score.
  • But if a word appears in many documents, it's not a unique identifier. Give the word a low score.
MAKING PREDICTIONS

- **Customers** profile
- **Items** profile

$$U(C,I) = \cos(\alpha) = \frac{C \cdot I}{\|C\| \|I\|}$$

- Cosine distance is the angle $\alpha$ and cosine similarity is $180 - \alpha$
- For mathematical convenience we use $\cos(\alpha)$ as the similarity measure and call it cosine similarity
PROS AND CONS

- Doesn’t need data about other uses
- Works good with unique tastes
- Solves cold start problem (items)
- The approach is interpretable
- Finding relevant features is not easy
- Doesn’t capture multiple interest
- Doesn’t consider popular items for similar users
- Cold start problem (users)
```python
def content_model(self):
    petition_cosine_similarities = linear_kernel(np.array(self.petitionF_lists))
    count=0
    results = {}
    resultsIndexes={}
    for idx, row in self.petitionFSorted.iteritems():
        similar_indices = petition_cosine_similarities[count].argsort()[::-10:-1]
        similar_items = [(petition_cosine_similarities[count][i], self.petitionFSorted[idx]) for i in similar_indices]
        # First item is the item itself, so remove it.
        # Each dictionary entry is like: [(1,2), (3,4)], with each tuple being (score, item_id)
        results[idx] = similar_items[1:]
        resultsIndexes[idx]=similar_indices[1:]
        count+=1
    print('done!')
    self.pResults=results
    self.pResultsIndexes=resultsIndexes
```
USER TO USER COLLABORATIVE FILTERING

- Consider Customers C
- Find set N of other customers whose ratings are similar to C’s rating
- Predict C’s rating based on ratings of users in N
- We need to define a notion of similarity between customers
EXAMPLE

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
<th>Hard</th>
<th>BNH</th>
<th>Twill</th>
</tr>
</thead>
<tbody>
<tr>
<td>James</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Megan</td>
<td>5</td>
<td>5</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Michael</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Vincent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

Consider users $x$ & $y$ with ratings $r_x$ & $r_y$
Define a similarity metric $\text{sim}(x,y)$
### JACCARD SIMILARITY

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
<th>Hard</th>
<th>BNH</th>
<th>Twill</th>
</tr>
</thead>
<tbody>
<tr>
<td>James</td>
<td>4</td>
<td></td>
<td></td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Megan</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Michael</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Vincent</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

\[
\text{Sim}(x,y) = \frac{|r_x \cap r_y|}{|r_x \cup r_y|}
\]

\[A = \text{Sim}(\text{James}, \text{Megan}) = \frac{1}{5}\]

\[B = \text{Sim}(\text{James}, \text{Michael}) = \frac{2}{4}\]

A < B, Not Intuitive … Doesn’t capture the ratings

*Mining massive data sets chapter 9*
COSINE SIMILARITY

\[ \text{Sim}(x,y) = \cos(r_x, r_y) \]

\[ A = \text{Sim}(\text{James, Megan}) = 0.38 \]

\[ B = \text{Sim}(\text{James, Michael}) = 0.32 \]

A > B … treats missing ratings as negative (problem!)

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
<th>Hard</th>
<th>BNH</th>
<th>Twill</th>
</tr>
</thead>
<tbody>
<tr>
<td>James</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Megan</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Michael</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Vincent</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>


## CENTERED COSINE SIMILARITY

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
<th>Hard</th>
<th>BNH</th>
<th>Twill</th>
</tr>
</thead>
<tbody>
<tr>
<td>James</td>
<td>4</td>
<td></td>
<td></td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Megan</td>
<td>5</td>
<td>5</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Michael</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Vincent</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

Normalize ratings by subtracting the row mean
### CENTERED COSINE SIMILARITY

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
<th>Hard</th>
<th>BNH</th>
<th>Twill</th>
</tr>
</thead>
<tbody>
<tr>
<td>James</td>
<td>2/3</td>
<td>0</td>
<td>0</td>
<td>5/3</td>
<td>-7/3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Megan</td>
<td>1/3</td>
<td>1/3</td>
<td>-2/3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Michael</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-5/3</td>
<td>1/3</td>
<td>4/3</td>
<td>0</td>
</tr>
<tr>
<td>Vincent</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- $\text{Sim}(x,y) = \cos(r_x, r_y)$
- $A = \text{Sim}(\text{James, Megan}) = 0.09$
- $B = \text{Sim}(\text{James, Michael}) = -0.56$
- $A > B$, More intuitive
- Handle “Tough raters” & “Easy raters”
- Also named as **Pearson Correlation**

Mining massive data sets chapter 9
RATING PREDICTION

• Let \( r_x \) vector of users \( c \)'s ratings

• Let \( N \) be the set of \( k \) users most similar to \( c \) who have also rated item \( I \)

• Prediction for user \( c \) and item \( I \)

• Simple approach: \( r_{xi} = \frac{1}{K} \sum_{y \in N} r_{yi} \)

• Weighted average: \( r_{xi} = \frac{\sum_{y \in N} s_{xy} r_{yi}}{\sum_{y \in N} s_{xy}} \)

  • \( s_{xy} = \text{sim}(x,y) \)
ITEM TO ITEM COLLABORATIVE FILTERING

- Same approach
  - For item I find other similar items
  - Predict rating for item I based on ratings for similar items
  - Similarity metrics and prediction function can be the same as in user-user

\[ r_{xi} = \frac{\sum_{y \in N(i;x)} S_{ij} r_{yi}}{\sum_{y \in N(i;x)} S_{ij}} \]

- \( S_{ij} \) similarity of item I & j
- \( r_{yi} \) similarity of customer x to item j
- \( N(i;x) \) set items rated by x similar to j
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>5</td>
<td></td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>?</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>
We use Pearson correlation (centered-cosine similarity)
The 2 nearest neighborhood to item I is Items 3 & 6

\[ r_{15} = \frac{(0.41 \times 2 + 0.59 \times 3)}{(0.41 + 0.59)} = 2.6 \]
ITEM-ITEM VS USER-USER

• Theoretically, dual approaches and should have same performance

• Practically, item-item outperforms in most cases

• Reason … Items are simpler than people!
  • Items can belong to a small subset that belongs to while users may have varied tastes
  • Items similarity are meaningful than user similarity
PROS AND CONS

• Doesn’t need feature selection
• Cold start problem
  • New users in the system
• Sparsity
  • User/ratings matrix is sparse
• First raters:
  • Can’t recommend unrated items
• Popularity bias:
  • Tends to recommend popular items
LATENT FACTOR MODELS

• Adopt Machine Learning, where we use optimization to build a better recommender

• Most famous is Matrix factorization

• Uses dimensionality reduction
WE NEED TO GO DEEPER
DETOUR – CURSE OF DIMENSIONALITY

- Goal is to find underlying distribution
- As dimension increase you need exponentially more data to find the distribution

WHY DIMENSIONALITY REDUCTION

• Remove the noise and have better signal

• Visualization (t-SNE, UMAP)

• Memory and processing
HOW TO REDUCE DIMENSIONALITY

- PCA
- LDA
- GDA
- Autoencoder
- t-SNE
- UMAP
- SVD
SVD

- Rank of a matrix: Number of linearly independent columns
  - Once we know it, we can re-write the matrix more efficient as a linear combination
- Goal is to do discover the axis of the data …
  - Rather than representing a point as 2 coordinates, we represent it as 1 coordinate
- It minimized the re-construction error (SSE)
$\mathbf{A}_{[I \times C]} = \mathbf{U}_{[I \times r]} \mathbf{\Sigma}_{[r \times r]} \mathbf{(V}_{[C \times r]} \mathbf{)^T}$

- **Input Matrix**
- **Unique, and orthogonal matrices**
- **Diagonal matrix represents the weights**
What happens when we multiply vectors in this circle by $X$?

1- The coordinate axis get rotated
2- the new axis get elongated
3- The ellipse gets rotated
SVD

\[ A = \Sigma U \]

\[ B = \begin{array}{c}
\Sigma \\
uT
\end{array} \]

B is the best approximation to A
MATRIX FACTORIZATION

- Rating matrix $R = Q \cdot P^T$

Where $Q$ is items to $K$ (Thin and long) and $P^T$ is $K$ to users (Fat and small)
### DETOUR - EVALUATION

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>?</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>4</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

### Root-mean-square-error (RMSE)

\[
\sqrt{\frac{\sum_{i \in T} (r_{xi} - r_{xi}^*)^2}{N}}
\]

Where \( N = |T| \)
- \( r_{xi} \) is the predicted rating
- \( r_{xi}^* \) is the actual rating

Not the best MOP, but this will be out of the scope of the tutorial.
Other Alternatives are precision user's top K

Test Set
How to estimate the missing ratings of user $x$ to item $i$?

- $r_{xi} = Q_i \cdot P^T_x$

Objective is to minimize the reconstruction error

$$\frac{1}{|R|} \sqrt{\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2}$$
SVD IS AWESOME!

• SVD minimized reconstruction error (SSE)
  
  \[
  \min_{U,V,\Sigma} \sum_{i,j \in A} (A_{ij} - [U \Sigma V^T]_{ij})^2
  \]

• This also minimizes RMSE

• Problem is that SVD requires a dense matrix, while our utility matrix is sparse
• We minimize the SSE on the training data, an implication of that is the model tends to memorize the data and fit to noise
• Doesn’t generalize well for the test data
• We bump into overfitting problem
• Use cross validation and regularization to prevent overfitting
WORK FLOW

• We are not interested in the absolute value of the objective function but rather the values of P and Q that minimized the objective function

• A good starting point is to initialize P and Q using SVD

• Use any optimization techniques to solve the quadratic equation
  • GD, SDG, Hessian, LBFGS, Liblinear

\[
\min_{P, Q} \sum_{\text{training}} (r_{xi} - q_i p_x)^2 + \left[ \lambda_1 \sum_x \| p_x \|_2^2 + \lambda_2 \sum_i \| q_i \|_2^2 \right]
\]
def CF_MF_recommender(self):
    lambda = 0.1  # Regularisation weight
    k = 20       # Dimensionality of the latent feature space
    m, n = self.R.shape  # Number of users and items
    n_epochs = 100     # Number of epochs
    gamma = 0.01       # Learning rate

    # unknown user and items features
    P = np.random.rand(m,k)  # initial user feature matrix with random numbers
    Q = np.random.rand(n,k)  # initial petition feature matrix with random numbers
    train_errors = []
    test_errors = []
    # Only consider non-zero matrix
    users, items = self.R.nonzero()
    for epoch in range(n_epochs):
        for u, i in zip(users, items):
            e = self.R[u, i] - self.prediction(P[u,:], Q[i,:])  # Calculate error for gradient
            P[u,:] += gamma * (e * Q[i,:] - lambda * P[u,:])  # Update latent user feature matrix
            Q[i,:] += gamma * (e * P[u,:] - lambda * Q[i,:])  # Update latent petition feature matrix
        train_rmse = self.rmse(self.I, self.R, Q, P)   # Calculate root mean squared error from train dataset
        test_rmse = self.rmse(self.I2, self.T, Q, P)    # Calculate root mean squared error from test dataset
        train_errors.append(train_rmse)
        test_errors.append(test_rmse)
    self.MF_RStar=self.MatrixPred(P,Q)
HYBRID METHODS

- Add content-based to collaborative filtering

- Implement two or more recommender and combine predictions (ensemble of models)

- This will be discussed in the case study
CASE STUDY

Don’t put flame retardant chemicals in sports drinks!

Sarah Kavanagh started this petition to Gatorade and 8 others

The other day, I Googled “brominated vegetable oil.” It was the

Confirmed victory

This petition made change with 205,465 supporters!

Gatorade: Don’t put flame retardant chemicals in sports drinks!

Share on Facebook

Send a Facebook message

Send an email to friends

Tweet to your followers

Copy link
climate change

Petition to: Michael Gove
Keep Climate Change in the Curriculum

... inspired me to get out there and do as much as I could. Climate change is the most pressing and climate change themselves. But to obscure the truth, and any chance we have of acting from children... change education for under 14s. We must keep climate change in our curriculum in order. Read more

Baba Maarsha
House of Commons, United Kingdom
At: 36,348 supporters
Created: Mar 10, 2013

Petition to: Donald Trump
Tell Trump To #ActOnClimate

...President-elect Trump has called climate change a Chinese hoax, vowed to dismantle Americas... climate change threatens Americas economy, national security, and public health and safety. Thats why... written an open letter (read here) urging Donald Trump to take 6 key steps to. Read more

Climate Felon: WA
At: 164,411 supporters
Created: Dec 5, 2016

Call on Leaders to Act: Combat Climate Change Now

Climate change may be the most important challenge humanity has ever faced. The Paris Agreement... steps to prevent catastrophic climate change and ensure a cleaner, safer planet for future generations... cannot afford to wait any longer to cut harmful carbon emissions and combat. Read more

The Nature Conservancy
At: 186,017 supporters
Created: Nov 16, 2016
HOW CAN YOU IMPROVE THE RECOMMENDATIONS?

Elnoshokaty, & Wu, 2018
DESIGN SCIENCE APPROACH
EXAMPLE – LATENT SUB TOPIC FEATURES

The Graphical model of LDA (Blei et al., 2003)
THANK YOU!
REFERENCES

• **Mining of Massive Datasets** Jure Leskovec, Anand Rajaraman, Jeff Ullman

• **Performance of recommender algorithms on top-n recommendation tasks**—2010, by Paolo Cremonesi, Yehuda Koren, Roberto Turpin

• **Trust-aware recommender systems**—2007, by Paolo Massa, Paolo Avesani

• **A matrix factorization technique with trust propagation for recommendation in social networks**—2010, by Mohsen Jamali, Martin Ester

• **Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering**—2010, by Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, Nuria Oliver

• **Hidden factors and hidden topics: understanding rating dimensions with review text**—2013, by Julian McAuley, Jure Leskovec