PART 1. LARGE SCALE DATA ANALYTICS

4. RECOMMENDATION SYSTEMS

Sangmi Lee Pallickara
Computer Science, Colorado State University
http://www.cs.colostate.edu/~cs435

FAQs

• PA2

• Gradient descent algorithm and distributed implementations
  • Parameter server (TensorFlow)
  • Spark Mllib
Today’s topics

• Recommendation Systems
  • Overview
  • Background: Data Similarity
  • Amazon’s Item-to-Item recommendation system

Part 1. Large Scale Data Analytics
4. Recommendation Systems
   Introduction
This material is built based on


• Sandy Ryza, Uri Laserson, Sean Owen, and Josh Wills, *Advanced Analytics with Spark*, O’Reilly, 2015

“What percentage of the top 10,000 titles in any online media store (Netflix, iTunes, Amazon, or any other) will rent or sell at least once a month?”
The long tail phenomenon [1/2]

- Distribution of numbers with a portion that has a large number of occurrences far from the “head” or central part of the distribution
  - The vertical axis represents popularity
  - The items are ordered on the horizontal axis according to their popularity
  - The long-tail phenomenon forces online institutions to recommend items to individual users


The long tail phenomenon [2/2]

- “Touching the Void”, Joe Simpson, 1988
- “Into Thin Air: A Personal Account of the Mt. Everest Disaster”, Jon Krakauer, 1997
Recommendation systems

• Seek to predict the “rating” or “preference” that a user would give to an item

Applications of Recommendation Systems

• **Product recommendations**
  • Amazon or similar online vendors

• **Movie recommendations**
  • Netflix offers its customers recommendations of movies they might like

• **News articles**
  • News services have attempted to identify articles of interest to readers based on the articles that they have read in the past
  • Blogs, YouTube
Netflix Prize

• The Netflix Prize challenge concerned recommender systems for movies (October, 2006)

• Netflix released a training set consisting of data from almost 500,000 customers and their ratings on 18,000 movies.
  • More than 100 million ratings

• The task was to use these data to build a model to predict ratings for a hold-out set of 3 million ratings

Large Scale Data Analytics
4. Recommendation Systems
Background: Data Similarity
Jaccard Coefficient (a. without description)

• Compare two sets $P$ and $Q$ with the following formula:

$$StringJaccard(P, Q) = \frac{|P \cap Q|}{|P \cup Q|}$$

• Measures the fraction of the data that is shared between $P$ and $Q$
• Compared to all data available in the union of these two sets.
• What are $P$ and $Q$?
  • Set of tokens from Strings
  • Complete description about data (candidates)

Example

• $StringJaccard$

$S(c_1) = \{Thomas, Sean, Connery\}$
$S(c_2) = \{Sir, Sean, Connery\}$

What is the string Jaccard coefficient between $c1$ and $c2$?

$$StringJaccard(P, Q) = \frac{|P \cap Q|}{|P \cup Q|} = \frac{2}{4} = 0.5$$
Jaccard Coefficient (b. with description)

• Given two candidates, the Jaccard coefficient of two candidates $c_1$ and $c_2$ is given by,

\[
\text{DescriptionJaccard}(c_1, c_2) = \frac{|\text{OD}(c_1) \cap \text{OD}(c_2)|}{|\text{OD}(c_1) \cup \text{OD}(c_2)|}
\]

Example

• Now, specify the parts of a person’s name as
  • title, firstname, middlename, and lastname

  \[
  \text{OD}(c_1) = \{(\text{firstname}, \text{Thomas}), (\text{middlename}, \text{Sean}), (\text{lastname}, \text{Connery})\}
  \]

  \[
  \text{OD}(c_2) = \{(\text{title}, \text{Sir}), (\text{middlename}, \text{Sean}), (\text{lastname}, \text{Connery})\}
  \]

  \[
  \text{DescriptionJaccard}(c_1, c_2) = \frac{2}{4}
  \]
Question

• *DescriptionJaccard* and *StringJaccard* have the same value. Is this always true?

Example

• **What if “Sean” would have been put in the firstname/middlename attribute?**

\[
OD(c_1) = \{(middlename, Thomas), (firstname, Sean), (lastname, Connery)\}
\]
\[
OD(c_2) = \{(title, Sir), (middlename, Sean), (lastname, Connery)\}
\]

\[
DescriptionJaccard(c_1, c_2) = \frac{1}{5}
\]
Deficiencies of the Jaccard Similarity

• Some attribute is more descriptive
  • Title is less descriptive than firstname and lastname

• Very sensitive to typographical errors in single tokens
  • Shean Conery and Sean Connery have a similarity of zero.

Cosine similarity

• Given two $n$-dimensional vectors $V$ and $W$, the cosine similarity computes the cosine of the angle $\alpha$ between these two vectors.
Cosine similarity - Continued

• Given two \( n \)-dimensional vectors \( V \) and \( W \), the \textit{cosine similarity} computes \textit{the cosine of the angle} \( \alpha \) \textit{between these two vectors} as

\[
\text{CosineSimilarity}(V,W) = \cos(\alpha) = \frac{V \cdot W}{\|V\| \times \|W\|}
\]

Where \( \|V\| \) is the length of the vector \( V = [a,b,c,..] \) computed as

\[
\sqrt{a^2 + b^2 + c^2 + ...}
\]

Cosine similarity - Continued

• The vectors \( V \) and \( W \)
  • Tokens in a string
  • Descriptions of a candidate

• The \textit{d dimensions} of these vectors correspond to all \textit{d distinct tokens} in a set of strings.
  • Denoted as \( D \)

• For a large database, \( d \) may be large
  • \( V \) and \( W \) have high dimensionality \( d \)
Cosine similarity - Continued

If the degree between the vectors is 0
\[ \text{Cosine similarity: 1} \]

If the vectors are orthogonal
\[ \text{Cosine similarity: 0} \]

If the vectors are opposite
\[ \text{Cosine similarity: -1} \]

Cosine similarity - Continued

• The pair of breeds with highest cosine similarity?

• Preserving the “ratio” between weights
Weight of Token

• Vector contains a \textbf{weight} for each of the \( d \) distinct tokens

• How to measure the weight?
  • Measuring frequency
  • Term frequency – inverse document frequency (tf-idf)

Term frequency

• \textbf{Number of times} that term \( t \) occurs in the document \( d \)
  • Raw term frequency: \( f_{t,d} \)
  • Boolean “frequencies” \( tf(t,d) \) is 1 if \( t \) occurs in \( d \) and 0 otherwise
  • e.g. Logarithmically scaled frequency:
    \[
    tf(t,d) = 1 + \log f_{t,d}
    \]
    • 0 if \( f_{t,d} \) is zero
Example: What is the **raw** $tf^{"American", C3}$?

<table>
<thead>
<tr>
<th>CID</th>
<th>Name</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Allstate</td>
<td>Allstate</td>
</tr>
<tr>
<td>C2</td>
<td>American Automobile Association</td>
<td>American</td>
</tr>
<tr>
<td>C3</td>
<td>American National Insurance Company</td>
<td>Automobile Association</td>
</tr>
<tr>
<td>C4</td>
<td>Farmers Insurance</td>
<td>National Insurance</td>
</tr>
<tr>
<td>C5</td>
<td>GEICO</td>
<td>Company</td>
</tr>
<tr>
<td>C6</td>
<td>John Hancock Insurance</td>
<td>Farmers</td>
</tr>
<tr>
<td>C7</td>
<td>Liberty Insurance</td>
<td>GEICO</td>
</tr>
<tr>
<td>C8</td>
<td>Mutual Insurance of American Life Insurance</td>
<td>John Hancock</td>
</tr>
<tr>
<td>C9</td>
<td>Safeway Insurance Group</td>
<td>Liberty</td>
</tr>
<tr>
<td>C10</td>
<td>Westfield</td>
<td>...</td>
</tr>
</tbody>
</table>

$tf^{"American", C3} = 1$
Inverse document frequency

• Assigns higher weights to tokens that occurs less frequently in the scope of all candidate descriptions

\[ idf(t, D) = \log_{10} \frac{N}{|\{d \in D : t \in d\}|} \]

• Where, \( N \) is the total number of documents in the corpus
• Number of documents where the term \( t \) appears (i.e. \( tf(t,d) \neq 0 \))

Example: What is the \( idf_{American}, D \)? (D: this corpus)

<table>
<thead>
<tr>
<th>CID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Allstate</td>
</tr>
<tr>
<td>C2</td>
<td>American Automobile Association</td>
</tr>
<tr>
<td>C3</td>
<td>American National Insurance Company</td>
</tr>
<tr>
<td>C4</td>
<td>Farmers Insurance</td>
</tr>
<tr>
<td>C5</td>
<td>GEICO</td>
</tr>
<tr>
<td>C6</td>
<td>John Hancock Insurance</td>
</tr>
<tr>
<td>C7</td>
<td>Liberty Insurance</td>
</tr>
<tr>
<td>C8</td>
<td>Mutual Insurance of American Life Insurance</td>
</tr>
<tr>
<td>C9</td>
<td>Safeway Insurance Group</td>
</tr>
<tr>
<td>C10</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Allstate</td>
<td></td>
</tr>
<tr>
<td>American</td>
<td></td>
</tr>
<tr>
<td>Automobile</td>
<td></td>
</tr>
<tr>
<td>Association</td>
<td></td>
</tr>
<tr>
<td>National</td>
<td></td>
</tr>
<tr>
<td>Insurance</td>
<td></td>
</tr>
<tr>
<td>Company</td>
<td></td>
</tr>
<tr>
<td>Farmers</td>
<td></td>
</tr>
<tr>
<td>GEICO</td>
<td></td>
</tr>
<tr>
<td>John</td>
<td></td>
</tr>
<tr>
<td>Hancock</td>
<td></td>
</tr>
<tr>
<td>Liberty</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Example: What is the $idf_{"American", D}$?

<table>
<thead>
<tr>
<th>CID</th>
<th>Name</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Allstate</td>
<td>Allstate</td>
</tr>
<tr>
<td>C2</td>
<td>American Automobile Association</td>
<td>American</td>
</tr>
<tr>
<td>C3</td>
<td>American National Insurance Company</td>
<td>Automobile</td>
</tr>
<tr>
<td>C4</td>
<td>Farmers Insurance</td>
<td>Association</td>
</tr>
<tr>
<td>C5</td>
<td>GEICO</td>
<td>National</td>
</tr>
<tr>
<td>C6</td>
<td>John Hancock Insurance</td>
<td>Insurance</td>
</tr>
<tr>
<td>C7</td>
<td>Liberty Insurance</td>
<td>Company</td>
</tr>
<tr>
<td>C8</td>
<td>Mutual Insurance of American Life Insurance</td>
<td>Farmers</td>
</tr>
<tr>
<td>C9</td>
<td>Safeway Insurance Group</td>
<td>GEICO</td>
</tr>
<tr>
<td>C10</td>
<td>Westfield</td>
<td>John Hancock</td>
</tr>
</tbody>
</table>

$idf_{"American", D} = \log(10/3)$

$tf-idf$ weighting

- The product of its $tf$ weight and its $idf$ weight

$$W_{t,d} = (1 + \log_{10} tf_{t,d}) \times \log_{10}(N / df_t)$$

*For the total number of documents, $N$*

- If the numbers of terms are different in the documents, you should normalize $tf_{t,d}$ to $(tf_{t,d} / \text{(number of words in $d$)})$

- Best known weighting scheme in information retrieval
  - Note: the “-” in $tf-idf$ is a hyphen, not a minus sign!
  - Alternative names: tf.idf, tf $\times$ idf

- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection
### Example: What is the **tf-idf**","American","C3,D and tf-idf","American","C8,D?**

<table>
<thead>
<tr>
<th>CID</th>
<th>Name</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Allstate</td>
<td>Allstate</td>
</tr>
<tr>
<td>C2</td>
<td>American Automobile Association</td>
<td>American</td>
</tr>
<tr>
<td>C3</td>
<td>American National Insurance Company</td>
<td>Automobile</td>
</tr>
<tr>
<td>C4</td>
<td>Farmers Insurance</td>
<td>Association</td>
</tr>
<tr>
<td>C5</td>
<td>GEICO</td>
<td>National</td>
</tr>
<tr>
<td>C6</td>
<td>John Hancock Insurance</td>
<td>Insurance</td>
</tr>
<tr>
<td>C7</td>
<td>Liberty Insurance</td>
<td>Company</td>
</tr>
<tr>
<td>C8</td>
<td>Mutual Insurance of American Life Insurance</td>
<td>Farmers</td>
</tr>
<tr>
<td>C9</td>
<td>Safeway Insurance Group</td>
<td>GEICO</td>
</tr>
<tr>
<td>C10</td>
<td>Westfield</td>
<td>John</td>
</tr>
</tbody>
</table>

tf-idf","American","C3,D = \( (1+\log(1/4)) \times \log(10/3) \)

\( \text{tf-idf","American","C8,D = (1+\log(1/6)) \times \log(10/3) } \)
Example continued

- Compute the similarity between the two strings $s_1$="Farmers Insurance", $s_2$ = "Liberty Insurance"

- Term vector = (Allstate, American, Automobile, Association, National, Insurance, Farmers, Liberty, ...)

- $s_1$="Farmers Insurance" = (0, 0, 0, 0, 1, 1, 0 ...)
- $s_2$="Liberty Insurance" = (0, 0, 0, 0, 1, 0, 1, ...)

- Next step: find the weights

<table>
<thead>
<tr>
<th>CID</th>
<th>Name</th>
<th>Q</th>
<th>V</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>Allstate</td>
<td>Allstate</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c2</td>
<td>American Automobile Association</td>
<td>America</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c3</td>
<td>American National Insurance Company</td>
<td>Automobile</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c4</td>
<td>Farmers Insurance</td>
<td>Association</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c5</td>
<td>GEICO</td>
<td>National</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c6</td>
<td>John Hancock Insurance</td>
<td>Insurance</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>c7</td>
<td>Liberty Insurance</td>
<td>Company</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c8</td>
<td>Mutual Insurance of America Life Insurance</td>
<td>Farmers</td>
<td>😐</td>
<td>0</td>
</tr>
<tr>
<td>c9</td>
<td>Safeway Insurance Group</td>
<td>GEICO</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c10</td>
<td>Westfield</td>
<td>John</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hancock</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Liberty</td>
<td>😐</td>
<td></td>
</tr>
</tbody>
</table>
Example continued

- Compute the similarity between the two strings $s_1 = \text{“Farmers Insurance”}, s_2 = \text{“Liberty Insurance”}$

- Six of the candidates contain the token “Insurance”.
  - $idf_{\text{“Insurance”}}, D = \log_{10}(10/6) = 0.78$
  - $idf_{\text{“Farmers”}}, D = \log_{10}(10/1) = 1$
  - $idf_{\text{“Liberty”}}, D = \log_{10}(10/1) = 1$

- tf-idf $\text{“Farmers”, c}_4 = (1+\log_{10}1/2) \times \log_{10}(10/1) = 0.7$
- tf-idf $\text{“Insurance”, c}_4 = (1+\log_{10}1/2) \times \log_{10}(10/6) \approx 0.55$
- tf-idf $\text{“Liberty”, c}_7 = (1+\log_{10}1/2) \times \log_{10}(10/1) = 0.7$
- tf-idf $\text{“Insurance”, c}_7 = (1+\log_{10}1/2) \times \log_{10}(10/6) \approx 0.55$

<table>
<thead>
<tr>
<th>CID</th>
<th>Name</th>
<th>Q</th>
<th>V</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>Allstate</td>
<td>Allstate</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c2</td>
<td>American Automobile Association</td>
<td>America</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c3</td>
<td>American National Insurance Company</td>
<td>Automobil e</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c4</td>
<td>Farmers Insurance</td>
<td>Association</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c5</td>
<td>GEICO</td>
<td>National</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c6</td>
<td>John Hancock Insurance</td>
<td>Insurance</td>
<td>0.55</td>
<td>0.55</td>
</tr>
<tr>
<td>c7</td>
<td>Liberty Insurance</td>
<td>Company</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c8</td>
<td>Mutual Insurance of America Life Insurance</td>
<td>Farmers</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>c9</td>
<td>Safeway Insurance Group</td>
<td>GEICO</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c10</td>
<td>Westfield</td>
<td>John</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Example continued
Example continued

• Compute the similarity between the two strings $s_1=$ Farmers Insurance, $s_2 =$ Liberty Insurance

\[
\text{CosSimilarity}(V, W) = \cos(\alpha) = \frac{V \cdot W}{\|V\| \times \|W\|} = \frac{0.55 \times 0.55}{\sqrt{0.55^2 + 0.7^2}} \approx 0.38
\]

• What is the Jaccard similarity for the same case?

Large Scale Data Analytics
4. Recommendation Systems
Collaborative Filtering
Collaborative filtering

- Focus on the **similarity** of the user ratings for items
- Users are **similar if their vectors are close** according to some distance measure
  - E.g. Jaccard or cosine distance
- Collaborative filtering
  - The process of identifying similar users and recommending what similar users like

Measuring similarity

- How to measure similarity of users or items from their rows or columns in the utility matrix?
  - Jaccard Similarity for A and B: 1/5 (without considering the weight)
  - Jaccard Similarity for A and C: 2/4 (without considering the weight)

- For user A, user C **might have similar opinion** than user B
- Can user C provide a prediction for A?

<table>
<thead>
<tr>
<th></th>
<th>HP1</th>
<th>HP2</th>
<th>HP3</th>
<th>TW</th>
<th>SW1</th>
<th>SW2</th>
<th>SW3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>
Cosine similarity (1/2)

- We can treat blanks as a 0 values
- The cosine of the angle between A and B is
  \[
  \frac{4 \times 5}{\sqrt{4^2 + 5^2 + 1^2} \sqrt{5^2 + 5^2 + 4^2}} = 0.380
  \]

<table>
<thead>
<tr>
<th>HP1</th>
<th>HP2</th>
<th>HP3</th>
<th>TW</th>
<th>SW1</th>
<th>SW2</th>
<th>SW3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

Cosine similarity (2/2)

- We can treat blanks as 0 values
- The cosine of the angle between A and C is
  \[
  \frac{5 \times 2 + 1 \times 4}{\sqrt{4^2 + 5^2 + 1^2} \sqrt{2^2 + 4^2 + 5^2}} = 0.322
  \]

- A is slightly closer to B than to C
Normalizing ratings (1/2)

• Generous vs. Harsh Ratings
• What if we normalize ratings by subtracting from each rating the average rating of that user?
• Some rating (very low) will turn into negative numbers
• If we take the cosine distance, the opposite views of the movies will have vectors in almost opposite directions
  • It can be as far apart as possible

Normalizing ratings (2/2)

• The cosine of the angle between A and B

\[
\frac{(2/3) \times (1/3)}{\sqrt{(2/3)^2 + (5/3)^2 + (-7/3)^2} \sqrt{(1/3)^2 + (1/3)^2 + (-2/3)^2}} = 0.092
\]

<table>
<thead>
<tr>
<th></th>
<th>HP1</th>
<th>HP2</th>
<th>HP3</th>
<th>TW</th>
<th>SW1</th>
<th>SW2</th>
<th>SW3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2/3</td>
<td></td>
<td></td>
<td>5/3</td>
<td>-7/3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1/3</td>
<td>1/3</td>
<td></td>
<td>-2/3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>-5/3</td>
<td>1/3</td>
<td>4/3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
• The cosine of the angle between A and C

\[
\frac{(5/3) \times (-5/3) + (-7/3) \times (1/3)}{\sqrt{(2/3)^2 + (5/3)^2 + (-7/3)^2} \sqrt{(-5/3)^2 + (1/3)^2 + (4/3)^2}} = -0.559
\]

• A and C are much further apart than A and B.
• Neither pair is very close
• **A and C disagree on the two movies** they rated in common, while **A and B give similar scores** to the one movie they rated in common

<table>
<thead>
<tr>
<th></th>
<th>HP1</th>
<th>HP2</th>
<th>HP3</th>
<th>TW</th>
<th>SW1</th>
<th>SW2</th>
<th>SW3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2/3</td>
<td></td>
<td></td>
<td>5/3</td>
<td>-7/3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1/3</td>
<td>1/3</td>
<td>-2/3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td>-5/3</td>
<td>1/3</td>
<td>4/3</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

**Computational complexity**

• Worst case
  • \(O(MN)\) where \(M\) is the number of customers and \(N\) is the number of product catalog items
  • It examines \(M\) customers and up to \(N\) items for each customer
Computational complexity (1/3)

• The average customer vector is extremely sparse!
  • The algorithm’s performance tends to be closer to $O(M+N)$
  • Scanning every customer
    • $O(M)$ not $O(MN)$
    • Almost every customer has very small $N$
  • Few customers who have purchased or rated a significant percentage of items
    • Requires $O(N)$
    • 10 million customers and 1 million items?

Computational complexity (2/3)

• We can reduce $M$ by:
  • Randomly sampling the customers
  • Discarding customers with few purchases

• We can reduce $N$ by:
  • Discarding very popular or unpopular items
  • Partitioning the item space based on the product category or subject classification
Computational complexity (3/3)

- Dimensionality reduction techniques can reduce $M$ or $N$ by a large factor
  - Clustering
  - Principal component analysis

Disadvantage of space reduction

- Reduced recommendation quality
  - Sampled customer
    - More similar customers will be dropped
  - Item-space partitioning
    - It will restrict recommendations to a specific product or subject area
- Discarding most popular or unpopular items
  - They will never appear as recommendations
Large Scale Data Analytics

4. Recommendation Systems

Amazon.com: Item-to-item collaborative filtering

This material is built based on,

• Greg Linden, Brent Smith, and Jeremy York, “Amazon.com Recommendations, Item-to-Item Collaborative Filtering” IEEE Internet Computing, 2003
• Amazon.com uses recommendations as a targeted marketing tool
  • Email campaigns
  • Most of their web pages

Item-to-item collaborative filtering

• It does NOT match the user to similar customers

• Item-to-item collaborative filtering
  • Matches each of the user’s purchased and rated items to similar items
  • Combines those similar items into a recommendation list
Determining the most-similar match

• The algorithm builds a **similar-items table**
  • By finding items that **customers tend to purchase together**

• How about building a product-to-product matrix by iterating through all item pairs and computing a similarity metric for each pair?

• Many product pairs have no common customer
  • If you already bought a TV today, will you buy another TV again today?

Co-occurrence in a product-to-product matrix

• Suppose that there are 4 users A, B, C, D, and E with 5 products P1, P2, P3, P4, and P5
  • A purchased P1, P2, P3
  • B purchased P2, P3, P5
  • C purchased P3, P4
  • D purchased P2, P4
Co-occurrence in a product-to-product matrix

- Suppose that there are 4 users A, B, C, D, and E with 5 products p1, p2, p3, p4, and p5
- A purchased p1, p2, p3
- B purchased p2, p3, p5
- C purchased p3, p4
- D purchased p2, p4

<table>
<thead>
<tr>
<th></th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
<th>p5</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>p2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>p3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>p4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>p5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Similarity product-to-product matrix

- Similarity between products
  - P1: T-shirt
  - P2: video cable
  - P3: projector
  - P4: tv
  - P5: smart tv

- Assume that,
  - Similarity (P1, P2)=0.001
  - Similarity (P1, P3)=0.001
  - Similarity (P1, P4)=0.001
  - Similarity (P1, P5)=0.001
  - Similarity (P2, P3)=0.013
  - Similarity (P2, P4)=0.022
  - Similarity (P2, P5)=0.022
  - Similarity (P3, P4)=0.26
  - Similarity (P3, P5)=0.26
  - Similarity (P4, P5)=0.72

<table>
<thead>
<tr>
<th></th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
<th>p5</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>0</td>
<td>0.001</td>
<td>0.001</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>p2</td>
<td>0.001</td>
<td>0</td>
<td>0.013</td>
<td>0.022</td>
<td>0.022</td>
</tr>
<tr>
<td>p3</td>
<td>0.001</td>
<td>0.013</td>
<td>0</td>
<td>0.26</td>
<td>0</td>
</tr>
<tr>
<td>p4</td>
<td>0</td>
<td>0.022</td>
<td>0.26</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>p5</td>
<td>0</td>
<td>0.022</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Algorithm

- Calculating the similarity between a single product and all related products:

```
For each item in product catalog, P-a
   For each customer C who purchased P-a
      For each item P-b purchased by customer C
         Record that a customer purchased P-a and P-b
   For each item P-b
      Compute the similarity between P-a and P-b
```

Computing similarity

- Using cosine measure
  - Each vector corresponds to an item rather than a customer
  - $M$ dimensions correspond to customers who have purchased that item
Creating a similarity product-to-product table

• Similar-items table is extremely computing intensive
  • Offline computation
    • $O(N^2M)$ in the worst case
      • Where $N$ is the number of items and $M$ is the number of users
    • Average case is closer to $O(NM)$
      • Most customers have very few purchases
    • Sampling customers who purchase best-selling titles reduces runtime even more
      • With little reduction in quality

Scalability (1/2)

• Amazon.com has around 300 million customers and more than 562,382,292 cataloged items
• Traditional collaborative filtering does little or no offline computation
• Online computation scales with the number of customers and catalog items.
Scalability (2/2)

- Cluster models can perform much of the computation offline
  - Recommendation quality is relatively poor

- Content-based model
  - It cannot provide recommendations with interesting, targeted titles
  - Not scalable for customers with numerous purchases and ratings

Key scalability strategy for Amazon recommendations

- Creating the expensive similar-items table offline

- Online component
  - Looking up similar items for the user’s purchases and ratings
  - Scales independently of the catalog size or the total number of customers

- It is dependent only on how many titles the user has purchased or rated
Computing with MapReduce

Part 1

- For each item in product catalog, P-a
  - For each customer C who purchased P-a
    - For each item P-b purchased by customer C
      - Record that a customer purchased P-a and P-b

- For each item P-b
  - Compute the similarity between P-a and P-b

- A Mapper reads a customer’s purchase history
  - Extracts product id(s) and generates co-purchased pairs
  - Emits the combination of the product IDs (sorted) \( \rightarrow \) \((P1, P2), \text{“1”}\)
- A Reducer adds the occurrences of the co-purchased products
  - Emits the (a-pair-of-co-purchased-products, total count)

Part 2

- Cartesian product design pattern (see the join pattern)
- A pair of product vectors will be provided to a map function
  - E.g. \(P1(0, 1, 0, 0, 0, 2, 0), P2(0, 1, 1, 0, 0, 2, 3)\)
- Mapper will calculate the cosine similarity
  - Emits Cosine_Similarity \(((0, 1, 0, 0, 0, 2, 0), (0, 1, 1, 0, 0, 2, 3))\)
- No reducer required
Recommendation quality

• The algorithm recommends highly correlated similar items
  • Recommendation quality is excellent
  • Algorithm performs well with limited user data

Questions?