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

This material is built based on
• Sandy Rya, 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

- 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

- “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
Jaccard Coefficient (a. without description)

- Compare two sets $P$ and $Q$ with the following formula:
  \[ \text{StringJaccard}(P, Q) = \frac{|P \cap Q|}{|P \cup Q|} \]
- 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
  $S(c_1) = \{\text{Thomas, Sean, Connery}\}$
  $S(c_2) = \{\text{Sir, Sean, Connery}\}$
  What is the string Jaccard coefficient between $c_1$ and $c_2$?
  \[ \text{StringJaccard}(P, Q) = \frac{|P \cap Q|}{|P \cup Q|} = \frac{2}{4} = 0.5 \]

Jaccard Coefficient (b. with description)

- Given two candidates, the Jaccard coefficient of two candidates $c_1$ and $c_2$ is given by,
  \[ \text{DescriptionJaccard}(c_1, c_2) = \frac{|OD(c_1) \cap OD(c_2)|}{|OD(c_1) \cup OD(c_2)|} \]

Example

- Now, specify the parts of a person’s name as
  - title, firstname, middlename, and lastname
  $OD(c_1) = \{(\text{firstname}, \text{Thomas}), (\text{middlename}, \text{Sean}), (\text{lastname}, \text{Connery})\}$
  $OD(c_2) = \{(\text{title}, \text{Sir}), (\text{middlename}, \text{Sean}), (\text{lastname}, \text{Connery})\}$
  \[ \text{DescriptionJaccard}(c_1, c_2) = \frac{2}{4} \]

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?
  $OD(c_1) = \{(\text{middlename}, \text{Thomas}), (\text{firstname}, \text{Sean}), (\text{lastname}, \text{Connery})\}$
  $OD(c_2) = \{(\text{title}, \text{Sir}), (\text{middlename}, \text{Sean}), (\text{lastname}, \text{Connery})\}$
  \[ \text{DescriptionJaccard}(c_1, c_2) = \frac{1}{5} \]
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
  • Sean 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

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

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

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

If the degree between the vectors is 0
→ Cosine similarity: 1

If the vectors are orthogonal
→ Cosine similarity: 0

If the vectors are opposite
→ Cosine similarity: -1

• The pair of breeds with highest cosine similarity?
  • Preserving the "ratio" between weights
Weight of Token

- Vector contains a weight for each of the $d$ distinct tokens
- How to measure the weight?
  - Measuring frequency
  - Term frequency – inverse document frequency ($tf$-$idf$)

Example: What is the raw $tf$"American", $C_3$?

<table>
<thead>
<tr>
<th>CID</th>
<th>Name</th>
<th>$tf$ &quot;American&quot;, $C_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Allstate</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>American Automobile Association</td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>American National Insurance Company</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>Farmers Insurance</td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>GEICO</td>
<td></td>
</tr>
<tr>
<td>C6</td>
<td>John Hancock Insurance</td>
<td></td>
</tr>
<tr>
<td>C7</td>
<td>Liberty Insurance</td>
<td></td>
</tr>
<tr>
<td>C8</td>
<td>Mutual Insurance of American Life Insurance</td>
<td></td>
</tr>
<tr>
<td>C9</td>
<td>Safeway Insurance Group</td>
<td></td>
</tr>
<tr>
<td>C10</td>
<td>Westfield</td>
<td></td>
</tr>
</tbody>
</table>

Example: What is the $tf$"American", $C_3$?

<table>
<thead>
<tr>
<th>CID</th>
<th>Name</th>
<th>$tf$ &quot;American&quot;, $C_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Allstate</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>American Automobile Association</td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>American National Insurance Company</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>Farmers Insurance</td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>GEICO</td>
<td></td>
</tr>
<tr>
<td>C6</td>
<td>John Hancock Insurance</td>
<td></td>
</tr>
<tr>
<td>C7</td>
<td>Liberty Insurance</td>
<td></td>
</tr>
<tr>
<td>C8</td>
<td>Mutual Insurance of American Life Insurance</td>
<td></td>
</tr>
<tr>
<td>C9</td>
<td>Safeway Insurance Group</td>
<td></td>
</tr>
<tr>
<td>C10</td>
<td>Westfield</td>
<td></td>
</tr>
</tbody>
</table>

Example: What is the $idf$"American", $D$? ($D$: this corpus)

<table>
<thead>
<tr>
<th>CID</th>
<th>Name</th>
<th>$idf$ &quot;American&quot;, $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Allstate</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>American Automobile Association</td>
<td>1</td>
</tr>
<tr>
<td>C3</td>
<td>American National Insurance Company</td>
<td>1</td>
</tr>
<tr>
<td>C4</td>
<td>Farmers Insurance</td>
<td>1</td>
</tr>
<tr>
<td>C5</td>
<td>GEICO</td>
<td>1</td>
</tr>
<tr>
<td>C6</td>
<td>John Hancock Insurance</td>
<td>1</td>
</tr>
<tr>
<td>C7</td>
<td>Liberty Insurance</td>
<td>1</td>
</tr>
<tr>
<td>C8</td>
<td>Mutual Insurance of American Life Insurance</td>
<td></td>
</tr>
<tr>
<td>C9</td>
<td>Safeway Insurance Group</td>
<td>1</td>
</tr>
<tr>
<td>C10</td>
<td>Westfield</td>
<td>1</td>
</tr>
</tbody>
</table>

Inverse document frequency

- Assigns higher weights to tokens that occur less frequently in the scope of all candidate descriptions
  
  
  $idf(t, D) = \log_2 \left( \frac{N}{|D \in D : t \in d|} \right)$

- 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 continued

• Compute the similarity between the two strings s1="Farmers Insurance", s2 = "Liberty Insurance"

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

• s1="Farmers Insurance" = (0, 0, 0, 0, 0, 1, 1, 0, ...) 

• s2="Liberty Insurance" = (0, 0, 0, 0, 0, 0, 1, 0, 1, ...)

• Next step: find the weights

Example continued

• Example: What is the \( \text{idf} \) of "American", \( D \)?

\[
\text{idf}(\text{"American"}, \ D) = \log(10/3)
\]

• \( tf \) and \( idf \)

\[
W_{t,f} = (1 + \log tf) \times \log(N/df)
\]

For the total number of documents, \( N \)

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

• Best known weighting scheme in information retrieval

Alternative names: \( \text{tf-idf} \)

• Increases with the number of occurrences within a document

• Increases with the rarity of the term in the collection

\[
\text{tf-idf}(\text{"American"}, \ c_3, \ D) = (1 + \log 1/4) \times \log(10/3)
\]

Example continued

\[
\text{tf-idf}(\text{"American"}, \ c_3, \ D) = (1 + \log 1/4) \times \log(10/3)
\]

\[
\text{tf-idf}(\text{"American"}, \ c_3, \ D) = (1 + \log 1/6) \times \log(10/3)
\]
Example continued

- Compute the similarity between the two strings \( s_1 = "Farmers Insurance", \ s_2 = "Liberty Insurance" \)

- Six of the candidates contain the token "Insurance".
  - \( \text{idf "Insurance"}, D = \log_{10}(10/6) \approx 0.78 \)
  - \( \text{idf "Farmers"}, D = \log_{10}(10/1) = 1 \)
  - \( \text{idf "Liberty"}, D = \log_{10}(10/1) = 1 \)
- \( \text{tf-idf "Farmers"}, c_4 = (1 + \log_{10}(0.5)) \times \log_{10}(10/1) = 0.7 \)
- \( \text{tf-idf "Insurance"}, c_4 = (1 + \log_{10}(0.5)) \times \log_{10}(10/6) \approx 0.55 \)
- \( \text{tf-idf "Liberty"}, c_7 = (1 + \log_{10}(0.5)) \times \log_{10}(10/1) = 0.7 \)
- \( \text{tf-idf "Insurance"}, c_7 = (1 + \log_{10}(0.5)) \times \log_{10}(10/6) \approx 0.55 \)

<table>
<thead>
<tr>
<th>CID</th>
<th>Name</th>
<th>V</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Allstate</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>American Automobile Association</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>American National Insurance Company</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Farmers Insurance</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>GEICO</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>John Hancock Insurance</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Liberty Insurance</td>
<td>0</td>
<td>0.7</td>
</tr>
<tr>
<td>8</td>
<td>Mutual Insurance of America Life Insurance</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Safeway Insurance Group</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>Westfield</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Large Scale Data Analytics

4. Recommendation Systems

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?
Cosine similarity (1/2)

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

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></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 
  \[
  \cos(\theta) = \frac{5 \times 2 \times 1 \times 4 + 1 \times 4 + 5}{\sqrt{4^2 + 5^2 + 1^2 + 4^2 + 5^2}} = 0.322
  \]

- A is slightly closer to B than to C.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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
  \[
  \cos(\theta) = \frac{2/3 \times (1/3) + (2/3) \times (-1/3) + (-1/3) \times (2/3) + (-2/3) \times 0}{\sqrt{(2/3)^2 + (5/3)^2 + (-1/3)^2 + (-2/3)^2}} = 0.092
  \]

- 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 movie they rated in common.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2/3</td>
<td>1/3</td>
<td>-1/3</td>
<td></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.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2/3</td>
<td>1/3</td>
<td>-1/3</td>
<td></td>
</tr>
</tbody>
</table>

- O(MN)
- Worst case
- \(O(MN)\)
- Normalizing ratings
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
    - $O(M + O(MN))$
    - 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

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)=0.026
  - Similarity (P3, P4)=0.72
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

- 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) → ((P1, P2), “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, 0, 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?