FAQs

- How to improve the term project proposal?
  - Brainstorm with your teammates
  - Make an appointment with me (meeting must include all of the team members)

Today’s topics

- Linear Regression: Running with MapReduce
- Recommendation Systems
- Overview
- Background: Data Similarity

Fitting $h_\theta(x)$

\[
\begin{align*}
\theta_0 &= \theta_0 - \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) \\
\theta_1 &= \theta_1 - \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)x^i
\end{align*}
\]

Update $\theta_0$ and $\theta_1$ simultaneously

Gradient descent for Linear Regression

Repeat until convergence {
\[
\begin{align*}
\theta_0 &= \theta_0 - \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) \\
\theta_1 &= \theta_1 - \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)x^i
\end{align*}
\]

(for $j=0$ and $j=1$)
}

Part 1. Large Scale Data Analytics

3. Predictive Analysis

Linear Regression: Running with MapReduce
"Batch" Gradient Descent

- Batch
  - Each step of gradient descent uses all of the training examples

\[
\begin{align*}
\theta_0 &= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)}) \\
\theta_1 &= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})x^{(i)}
\end{align*}
\]

- Step 3. Calculate final results
  \[
  \theta_0^{\text{new}} = \theta_0 - \alpha \frac{1}{1,000} (\text{temp1} + \text{temp2} + \text{temp3} + \text{temp4})
  \]

- Step 1. 4 input splits
  \[
  \begin{align*}
  \text{temp1} &= \sum_{i=1}^{500} (h(x^{(i)}) - y^{(i)}) \\
  \text{temp2} &= \sum_{i=501}^{1000} (h(x^{(i)}) - y^{(i)}) \\
  \text{temp3} &= \sum_{i=1001}^{1500} (h(x^{(i)}) - y^{(i)}) \\
  \text{temp4} &= \sum_{i=1501}^{2000} (h(x^{(i)}) - y^{(i)})
  \end{align*}
  \]

Step 2. Calculate temp1 ~ 4

Running with MapReduce

- For the sample size 1,000 (m = 1,000)
  - Batch gradient descent:
    \[
    \begin{align*}
    \theta_0 &= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)}) \\
    \theta_1 &= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})x^{(i)}
    \end{align*}
    \]
  - Using 4 machines

For \( \theta_0 \)

- Step 1. 4 input splits
  \[
  \begin{align*}
  \text{temp1} &= \sum_{i=1}^{500} (h(x^{(i)}) - y^{(i)}) \\
  \text{temp2} &= \sum_{i=501}^{1000} (h(x^{(i)}) - y^{(i)}) \\
  \text{temp3} &= \sum_{i=1001}^{1500} (h(x^{(i)}) - y^{(i)}) \\
  \text{temp4} &= \sum_{i=1501}^{2000} (h(x^{(i)}) - y^{(i)})
  \end{align*}
  \]

- Step 2. Calculate temp1 ~ 4
  \[
  \text{temp1} = \sum_{i=1}^{500} (h(x^{(i)}) - y^{(i)})x^{(i)} \quad \text{temp2} = \sum_{i=501}^{1000} (h(x^{(i)}) - y^{(i)})x^{(i)} \\
  \text{temp3} = \sum_{i=1001}^{1500} (h(x^{(i)}) - y^{(i)})x^{(i)} \quad \text{temp4} = \sum_{i=1501}^{2000} (h(x^{(i)}) - y^{(i)})x^{(i)}
  \]

- Step 3. Calculate final results
  \[
  \theta_0^{\text{new}} = \theta_0 - \alpha \frac{1}{1,000} (\text{temp1} + \text{temp2} + \text{temp3} + \text{temp4})
  \]

For \( \theta_1 \)

- Step 1. 4 input splits
  \[
  \begin{align*}
  \text{temp1} &= \sum_{i=1}^{500} (h(x^{(i)}) - y^{(i)}) \\
  \text{temp2} &= \sum_{i=501}^{1000} (h(x^{(i)}) - y^{(i)}) \\
  \text{temp3} &= \sum_{i=1001}^{1500} (h(x^{(i)}) - y^{(i)}) \\
  \text{temp4} &= \sum_{i=1501}^{2000} (h(x^{(i)}) - y^{(i)})
  \end{align*}
  \]

- Step 2. Calculate temp1 ~ 4
  \[
  \text{temp1} = \sum_{i=1}^{500} (h(x^{(i)}) - y^{(i)})x^{(i)} \quad \text{temp2} = \sum_{i=501}^{1000} (h(x^{(i)}) - y^{(i)})x^{(i)} \\
  \text{temp3} = \sum_{i=1001}^{1500} (h(x^{(i)}) - y^{(i)})x^{(i)} \quad \text{temp4} = \sum_{i=1501}^{2000} (h(x^{(i)}) - y^{(i)})x^{(i)}
  \]

- Step 3. Calculate final results
  \[
  \theta_1^{\text{new}} = \theta_1 - \alpha \frac{1}{1,000} (\text{temp1} + \text{temp2} + \text{temp3} + \text{temp4})
  \]

This material is built based on

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

Large Scale Data Analytics

4. Recommendation Systems
"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
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: 
  
  $$\text{StringJaccard}(P, Q) = \frac{|P \cap Q|}{|P \cup Q|}$$

- 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**
  
  $S(c_1) = \{\text{Thomas}, \text{Sean}, \text{Connery}\}$
  
  $S(c_2) = \{\text{Sir}, \text{Sean}, \text{Connery}\}$
  
  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
  
  $OD(c_1) = \{(\text{firstname}, \text{Thomas}), (\text{middlename}, \text{Sean}), (\text{lastname}, \text{Connery})\}$
  
  $OD(c_2) = \{(\text{title}, \text{Sir}), (\text{middlename}, \text{Sean}), (\text{lastname}, \text{Connery})\}$
  
  $$\text{DescriptionJaccard}(c_1, c_2) = 2/4$$

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?**

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

\[ \text{DescriptionJaccard}(c_1, c_2) = \frac{1}{5} \]

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 as

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

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 \( d \) dimensions of these vectors correspond to all \( d \) distinct tokens in a set of strings.
  - Denoted as \( D \)

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

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)

Term frequency

- Number of times that term \( t \) occurs in the document \( d \)
  - Raw term frequency, \( f_{t,d} \)
  - Boolean "frequencies" \( f(t,d) \) is 1 if \( t \) occurs in \( d \) and 0 otherwise
  - Logarithmically scaled frequency:
    \[ f(t,d) = 1 + \log f_{t,d} \]
    - \( 0 \) if \( f_{t,d} \) is zero
### Inverse document frequency

- Assigns higher weights to tokens that occurred less frequently in the scope of all candidate descriptions.

\[ \text{idf}(t, D) = \log\left(\frac{N}{\sum_{d \in D} f(t, d)}\right) \]

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

### Example: What is the raw \( \text{tf} \) “American”, \( C_3 \)?

<table>
<thead>
<tr>
<th>C3</th>
<th>Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Westfield</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Liberty Insurance</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Mutual Insurance of American Life Insurance</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>GEICO</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Farmers Insurance</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Liberty Insurance</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>American National Insurance Company</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>GEICO</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>John Hancock Insurance</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>American Automobile Association</td>
<td></td>
</tr>
</tbody>
</table>

### Example: What is the \( \text{idf} \) “American”, \( D \)? (D: this corpus)

<table>
<thead>
<tr>
<th>D</th>
<th>Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Westfield</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Liberty Insurance</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Mutual Insurance of American Life Insurance</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>GEICO</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Farmers Insurance</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Liberty Insurance</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>American National Insurance Company</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>GEICO</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>John Hancock Insurance</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>American Automobile Association</td>
<td></td>
</tr>
</tbody>
</table>

### Example: What is the \( \text{tf} \cdot \text{idf} \) “American”, \( P \)?

<table>
<thead>
<tr>
<th>P</th>
<th>Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Westfield</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Liberty Insurance</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Mutual Insurance of American Life Insurance</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>GEICO</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Farmers Insurance</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Liberty Insurance</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>American National Insurance Company</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>GEICO</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>John Hancock Insurance</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>American Automobile Association</td>
<td></td>
</tr>
</tbody>
</table>

### \( \text{tf} \cdot \text{idf} \) weighting

- The product of \( \text{tf} \) weight and its \( \text{idf} \) weight.

\[ W_{ij} = (1 + \log_{10} f_{ij}) \times \log_{10}(N / df) \]

- For the total number of documents, \( N \).
- If the numbers of terms are different in the documents, you should normalize \( f_{ij} \) to \( df_j \) (number of \( df \) of \( j \)).

- Best known weighting scheme in information retrieval:
  - Note: the “-” is \( \times \) a hyphen, not a minus sign.
  - Alternative name: \( \text{TF-IDF} \) or \( \text{IDF} \).

- Increases with the number of occurrences within a document.
- Increases with the rarity of the term in the collection.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>tf</th>
<th>idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Allstate</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>American Automobile Association</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>3</td>
<td>American National Insurance Company</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>4</td>
<td>Farmers Insurance</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>5</td>
<td>GEICO</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>6</td>
<td>John Hancock Insurance</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>7</td>
<td>Liberty Insurance</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>8</td>
<td>Mutual Insurance of American Life</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>9</td>
<td>Safeway Insurance Group</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>10</td>
<td>Westfield</td>
<td>1</td>
<td>1.0</td>
</tr>
</tbody>
</table>

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, 1, …)

• Next step: find the weights

Example continued

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

• 6 of the candidates contain the token “Insurance”.

• idf “Insurance”: \( \log(10/6) \approx 0.78 \)

• idf “Farmers”: \( \log(10/1) \approx 1 \)

• idf “Liberty”: \( \log(10/1) \approx 1 \)

• \( \text{tf-idf “Farmers”}= (1+\log(10/6)) \times \log(10/1) \approx 0.7 \)

• \( \text{tf-idf “Liberty”}= (1+\log(10/6)) \times \log(10/1) \approx 0.7 \)

• \( \text{tf-idf “Insurance”}= (1+\log(10/6)) \times \log(10/1) \approx 0.75 \)

Example continued

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

• 6 of the candidates contain the token “Insurance”.

• idf “Insurance”: \( \log(10/6) \approx 0.78 \)

• idf “Farmers”: \( \log(10/1) \approx 1 \)

• idf “Liberty”: \( \log(10/1) \approx 1 \)

• \( \text{tf-idf “Farmers”}= (1+\log(10/6)) \times \log(10/1) \approx 0.7 \)

• \( \text{tf-idf “Liberty”}= (1+\log(10/6)) \times \log(10/1) \approx 0.7 \)

• \( \text{tf-idf “Insurance”}= (1+\log(10/6)) \times \log(10/1) \approx 0.75 \)
Example continued

• Compute the similarity between the two strings \( s_1 = \text{Farmers Insurance}, \ s_2 = \text{Liberty Insurance} \)

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

• What is the Jaccard similarity for the same case?

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
• Jaccard Similarity for A and C: 2/4
• For user A, user C might have similar opinion than user B
• Can user C provide a prediction for A?

Collaborative filtering

Cosine similarity (1/2)

• We can treat blanks as a 0 values
• The cosine of the angle between A and B is

\[
\frac{4 \times 5 + 5 \times 5 + 5 \times 5 + 5}{\sqrt{4^2 + 5^2 + 5^2 + 5^2} \times \sqrt{4^2 + 5^2 + 5^2 + 5^2}} = 0.380
\]

Cosine similarity (2/2)

• We can treat blanks as 0 values
• The cosine of the angle between A and C is

\[
\frac{5 \times 2 + 2 \times 4 + 4 \times 5 + 5}{\sqrt{5^2 + 2^2 + 2^2 + 4^2} \times \sqrt{5^2 + 2^2 + 2^2 + 4^2}} = 0.322
\]

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

- 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