PART 1.
LARGE SCALE DATA ANALYSIS USING MAPREDUCE

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

Today’s topics
• FAQs
• Distance measures
• Link Analysis

FAQs
• Programming Assignment 1
  - Feb. 21st 5:00PM via Canvas
  - Individual submission
  - Please do not submit your output file!

Distance Measures

Distance measure
A distance measure over a space is a function \( d(x,y) \) that takes two points in the space as arguments and produces a real number that satisfies the following axioms:

A metric on a set \( X \) is a function (A.K.A. distance function) \( d: X \times X \to [0, \infty) \),
1. \( d(x,y) \geq 0 \) (no negative distance)
2. \( d(x,y) = 0 \) if and only if \( x = y \)
3. \( d(x,y) = d(y,x) \) (distance is symmetric)
4. \( d(x,y) \leq d(x,z) + d(z,y) \) (the triangle inequality)

Distance measures
- Euclidean distances
  \[ d([x_1, x_2, \ldots, x_n], [y_1, y_2, \ldots, y_n]) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \]
- Jaccard distance
  \[ d_{\text{Jaccard}}(x,y) = 1 - \text{Jaccard}_\text{SIM}(x,y) \]
- Cosine distance
  Degree between the vectors
- Hamming distance
  The number of components in which they differ
  - 10111 and 11107
What are these?

- Archie
- Veronica
- Infoseek
- Snap
- Direct Hit
- Lycos
- AltaVista
- Excite
- Yahoo
- Google

Goals

- Understanding large problem with unstructured data
- Applying analytics using MapReduce

Early Search Engines

- They worked by crawling the Web and listing the terms
  - Words or other strings of characters other than white space
  - In an inverted index
  - An inverted index is a data structure that makes it easy, given a term, to find (pointer to) all the places where that term occurs

Inverted index (1/2)

- Inverted index
  - For given texts:
    
    \[
    T[0] = \text{"Colorado State University"}
    T[1] = \text{"Colorado water source"}
    T[2] = \text{"University of Colorado"}
    \]
  - We have the following inverted file index

```

<table>
<thead>
<tr>
<th>Term</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Colorado&quot;:</td>
<td>{0,1,2}</td>
</tr>
<tr>
<td>&quot;water&quot;:</td>
<td>{1}</td>
</tr>
<tr>
<td>&quot;State&quot;:</td>
<td>{0}</td>
</tr>
<tr>
<td>&quot;University&quot;:</td>
<td>{0,2}</td>
</tr>
<tr>
<td>&quot;water&quot;:</td>
<td>{1}</td>
</tr>
</tbody>
</table>
```

Inverted index (2/2)

- A term search for the terms, "Colorado", "State", and "University" would give the set

\[
\{0,1,2\} \cap \{0\} \cap \{0,2\} = \{0\}
\]
Term spam
- If you were selling shirts on the Web
  - All you care about was that people would see your page
- You could add a term like "movie" to your page
  - Add thousands of times
    - It does not even need to show
    - Give the same color as background to the letters
- A search engine would think this page is very important one about "movie"
- You could go to the search engine and search "movie" and see the first listed page
  - Copy that page with the same color as background

Link Analysis
PageRank Algorithm

PageRank
- Goals
  - Providing effective summaries for the search results
  - Ordering/Ranking results
- Simulate random Web surfers
  - Pages that would have a large number of surfers were considered more "important" than pages that would rarely be visited
- The content of a page was judged not only by the terms appearing on that page
  - But by the terms used in or near the links to that page

Definition of PageRank
- A function that assigns a real number to each page in the Web
- The higher the PageRank of a page, the more "important" it is
- There is NOT one fixed algorithm for assignment of PageRank
Example [1/5]

- Page A has links to B, C and D
- Page B has links to A and D
- Page C has a link to A
- Page D has links to B and C

Example [2/5]

- Suppose that a random surfer starts at page A
- Page B, C and D will be the next with probability 1/3
  - 0 probability of being at A

Example [3/5]

- Now suppose the random surfer at B
  - B has probability ¼ of being at A, ¼ of being at D and 0 of being at B or C

Example [4/5]

- Transition matrix $M$
  - What happens to random surfers after one step
  - $M$ has $n$ rows and columns
  - What is the transition matrix for this example?
Example

\[
M = \begin{bmatrix}
0 & 1/2 & 1 & 0 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 1/2 & 0 & 0
\end{bmatrix}
\]

- The first column
  - a surfer at A has a 1/3 probability of next being at each of the other pages
- The second column
  - a surfer at B has a \( 1/2 \) probability of being next at A and the same for being at D

What does this matrix mean?

- If we surf at any of the n pages of the Web with equal probability
  - The initial vector \( v_0 \) will have \( 1/n \) for each component
  - If \( M \) is the transition matrix of the Web
    - After the first one step, the distribution of the surfer will be \( Mv_0 \)
    - After two steps, \( M(Mv_0) = M^2v_0 \) and so on

What does this matrix mean?

- The probability \( x_i \) that a random surfer will be at node \( i \) at the next step
  \[
x_i = \sum_j m_{ij}v_j
\]
- \( m_{ij} \) is the probability that a surfer at node \( j \) will move to node \( i \) at the next step
- \( v_j \) is the probability that the surfer was at node \( j \) at the previous step

What does this matrix mean?

- The distribution of the surfer approaches a limiting distribution \( v \) that satisfies \( v = Mv \) provided two conditions are met:
  1. The graph is strongly connected
     - It is possible to get from any node to any other node
  2. There are no dead ends
     - Nodes that have no arcs out

Note: This condition will be re-visited with taxation of the PageRank algorithm
What does this matrix mean? [6/6]

- The limit is reached when multiplying the distribution by \( M \) another time does not change the distribution
- The limiting \( v \) is an eigenvector of \( M \)
- For the Web, 50-75 iterations are sufficient to converge to within the error limits of double-precision arithmetic

Example

\[
M = \begin{bmatrix}
0 & 1/2 & 1 & 0 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 1/2 & 0 & 0 \\
\end{bmatrix}
\]

- Suppose we apply this process to the matrix \( M \)
- The initial vector \( v_0 \) and \( v_1 \) after multiplying \( M \)

\[
v_1 = Mv_0 = \begin{bmatrix}
0/24 \\
9/24 \\
5/24 \\
5/24 \\
\end{bmatrix}
\]

What is the \( v_2 \)?

\[
M = \begin{bmatrix}
0 & 1/2 & 1 & 0 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 1/2 & 0 & 0 \\
\end{bmatrix}
\]

- Suppose we apply this process to the matrix \( M \)
- The initial vector \( v_0 \) and \( v_1 \) after multiplying \( M \)

\[
v_1 = Mv_0 = \begin{bmatrix}
0/24 \\
9/24 \\
5/24 \\
5/24 \\
\end{bmatrix}
\]

Example continued

- The sequence of approximations to the limit
  - We get by multiplying at each step by \( M \) is

\[
\begin{array}{cccc}
1/4 & 9/24 & 55/48 & 11/32 \\
1/4 & 5/24 & 11/48 & 7/32 \\
1/4 & 5/24 & 11/48 & 17/32 \\
1/4 & 5/24 & 7/32 & 2/9 \\
\end{array}
\]

- This difference in probability is not noticeable
- In the real Web, there are billions of nodes of greatly varying importance
  - The probability of being at a node like www.amazon.com is orders of magnitude greater than others

Link Analysis
Matrix multiplication using MapReduce
Matrix-vector Multiplication by MapReduce [1/3]

- Suppose we have an \( n \times n \) matrix \( M \), whose element in row \( i \) and column \( j \) will be denoted \( M_{ij} \).
- Then the matrix-vector product is the vector \( x \) of length \( n \), whose \( i \)th element \( x_i \) is given by,

\[
x_i = \sum_{j=1}^{n} M_{ij}v_j
\]

Matrix-vector Multiplication by MapReduce [2/3]

- If \( n = 100 \), we do not need DFS or MapReduce.
- However, if this calculation is a part of ranking Web pages (\( n \) is 10M) that goes on at search engine? The vector \( v \) cannot fit in main memory.
  - More than 1.4B pages.

Matrix-vector Multiplication by MapReduce [3/3]

- The matrix \( M \) and the vector \( v \) each will be stored in a file of the DFS(HDFS).
- Assume that row-column coordinates of each matrix element will be discoverable.
  - Either from its position in the file or explicit coordinates.

The Map function

- The Map function is written to apply to one element of \( M \).
- Each Map task will operate on a chunk of the matrix \( M \).
- From each matrix element \( m_{ij} \), it produces the key-value pair \( (i, m_{ij}) \).
- All terms of the sum that make up the component \( x_i \) of the matrix-vector product will get the same key, \( i \).

The Reduce function

- Sums all the values associated with a given key \( i \).
- The result will be a pair \( (i, x_i) \).

If the vector \( v \) cannot fit in main memory?

- It is possible that the vector \( v \) is so large that it will not fit in its main memory entirely.
- We can divide the matrix into vertical stripes of equal width and divide the vector into an equal number of horizontal stripes of the same height.
  - The goal is to use enough stripes so that the portion of the vector in one stripe can fit into main memory.
Matrix $M$ and Initial Vector $v$ divided into five stripes.

The $i$th stripe of the matrix multiplies only components from the $i$th stripe of the initial vector.

Results:

\[
\begin{align*}
(0.002 & \times \frac{1}{n}) + (0.017 & \times \frac{1}{n}) + (0.003 & \times \frac{1}{n}) + (0.010 & \times \frac{1}{n}) + \cdots \\
+ (M_{01} & \times v_1) + (M_{02} & \times v_2) + \cdots \\
\end{align*}
\]

Structure of Web (1/3)
- Is the Web strongly connected?

Structure of Web (2/3)
- Consisting of pages reachable from the In Component but unable to reach the SCC
- Tendrils Out
- Tendrils In
- Strongly Connected Component

Structure of Web (3/3)
- Tubes
  - Pages reachable from the In component and able to reach the output component, but unable to reach the SCC
- Isolated components
  - Unreachable from the large components
### Anomalies from the Web structure

- These structures violate the assumptions needed for the Markov process iteration to converge to a limit
  - When a random surfer enters the out-component, they can never leave
  - Surfers starting in either the SCC or in-component are going to wind up in either the out-component or a tendril off the in-component
  - No page in the SCC or in-component winds up with any probability of a surfer being there
  - Nothing in the SCC or in-component will be of any importance

### Problems we need to avoid

- Dead end
  - A page that has no links out
  - Surfers reaching such a page will disappear
  - In the limit, no page that can reach a dead end can have any PageRank at all

- Spider traps
  - Groups of pages that all have outlinks but they never link to any other pages

### Avoiding Dead Ends

- If we allow dead ends
  - The transition matrix of the Web is no longer stochastic
    - Some of the columns will sum to 0 rather than 1

- If we compute $M^k$ for increasing powers of a substochastic matrix
  - Some of all of the components of the vector go to 0
  - substochastic matrix
    - A matrix whose column sums are at most 1
    - Importance "drains out" of the Web
    - No information about the relative importance of pages
Repeatedly multiplying the vector by $M$:

\[
\begin{bmatrix}
\frac{1}{6} & \frac{1}{24} & \frac{5}{48} & \frac{21}{288} & 0 \\
\frac{1}{6} & \frac{1}{24} & \frac{7}{48} & \frac{31}{288} & 0 \\
\frac{1}{6} & \frac{1}{24} & \frac{7}{48} & \frac{31}{288} & 0 \\
\frac{1}{6} & \frac{1}{24} & \frac{7}{48} & \frac{31}{288} & 0 \\
\end{bmatrix}
\]

- The probability of a surfer being anywhere goes to 0 as the number of steps increase.

Approaches to dealing with dead ends

- Recursive deletion
  - Drop the dead ends from the graph
  - Drop their incoming arcs as well
  - Doing so may create more dead ends
  - Drop those new dead ends

- Taxation
  - Modify the process by which random surfers are assumed to move about the Web

Example of recursive deletion (1/4)

Example of recursive deletion (2/4)

- The final matrix for the graph is

\[
M = \begin{bmatrix}
0 & \frac{1}{2} & 0 \\
\frac{1}{2} & 0 & 1 \\
\frac{1}{2} & \frac{1}{2} & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
\frac{1}{3} & \frac{1}{6} & \frac{3}{12} & \frac{5}{24} & \frac{2}{9} \\
\frac{1}{3} & \frac{3}{6} & \frac{5}{12} & \frac{11}{24} & \ldots & \frac{4}{9} \\
\frac{1}{3} & \frac{2}{6} & \frac{4}{12} & \frac{8}{24} & \ldots & \frac{3}{9} \\
\end{bmatrix}
\]

Example of recursive deletion (3/4)

- We still need to compute deleted nodes (C and E)
  - C was the last to be deleted
    - We know all its predecessors have PageRanks (A and D)
    - Therefore,
      \[
      \text{PageRank of C} = \frac{1}{3} \times \frac{2}{9} + \frac{1}{3} \times \frac{3}{9} = \frac{13}{54}
      \]

Example of recursive deletion (4/4)

- Now, we can compute the PageRank for E
  - Only one predecessor, C
    - The PageRank of E is the same as that of C (13/54)

- The sums of the PageRanks exceeds 1
  - It cannot represent the distribution of a random surfer
  - It provides a good estimate