Voting Pattern Project
Conclusion and Discussion

Feb. 13, 2014
Today’s Class

• Discuss the voting-pattern project results
• How much computation gets used when a problem gets larger?
  – Sorting numbers
• What have we learned so far?
• Presenting final projects: make a website in 5 minutes
• Project examples
• Clustering
Conclusion from last class

• What matters isn’t liberal vs conservative...
• It’s the network structure in the senate!
  – Who is in what group, what subgroup, etc.
  – Who are connectors between groups?
• Computation enabled this insight.
What about the House?

• Lots more members
• Lots less agreement
• SLOOOOWWWW!
  – How come? It’s only 4.3 times as large?
All that grouping had to be done by hand...

• Isn’t there some way to do that automatically?
  – CS53, clustering exercise
  – CS157, “Clustering”; esp. “Spectral Clustering”
As data grows, how does computation grow?

• Sorting a 13-card hand in bridge takes longer than sorting a 5-card poker hand
  – Does it take $13/5$ths as long? More? Less?

• More. But how much more?
  – Depends how you sort
Sorting: Bubble sort

• Start with a list and an empty answer-list
• Repeat...
  – Scan through the list to find smallest element
  – Move that element to end of answer list
• ...until the list is empty
• At that point, the answer-list contains the same items, sorted smallest-to-largest
Sorting: Bubble sort

- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8
- 4 1 2 5 3 6 0 8

0 ← Moved to “answer list”
<table>
<thead>
<tr>
<th></th>
<th>4 2 5 3 6 8</th>
<th>0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4 2 5 3 6 8</td>
<td>0 1</td>
</tr>
<tr>
<td></td>
<td>4 2 5 3 6 8</td>
<td>0 1</td>
</tr>
<tr>
<td></td>
<td>4 2 5 3 6 8</td>
<td>0 1</td>
</tr>
<tr>
<td></td>
<td>4 2 5 3 6 8</td>
<td>0 1</td>
</tr>
<tr>
<td></td>
<td>4 2 5 3 6 8</td>
<td>0 1</td>
</tr>
<tr>
<td></td>
<td>4 5 3 6 8</td>
<td>0 1 2</td>
</tr>
</tbody>
</table>
• 4 5 6 8 0 1 2 3
• 4 5 6 8 0 1 2 3
• 4 5 6 8 0 1 2 3
• 4 5 6 8 0 1 2 3
• 5 6 8 0 1 2 3 4
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 568
- 568
- 568
- 568
- 68
• 6 8 0 1 2 3 4 5
• 6 8 0 1 2 3 4 5
• 8 0 1 2 3 4 5 6
• 8

• 0 1 2 3 4 5 6

• 0 1 2 3 4 5 6 8

• Done!
For $k$ items...

- We did $k$ comparisons the first time
- We did $k - 1$ comparisons the second
- ...
- We did 1 comparison the last time
- Total: $T = 1 + 2 + 3 + \ldots + k$
How big is $T$?

- Draw a picture
How big is $T$

• Here’s $2T$:  

\[\text{...}\]
How big is \( T \)

- Here’s \( 2T \):
How big is $T$

- Here’s $2T$:
- So $2T = k(k+1) = k^2 + k$
- $T = \frac{1}{2} (k^2 + k)$
\[ T = \frac{1}{2} (k^2 + k) \]

• If you double the size of the problem \( (k) \), the work done \( (T) \) multiplies by a little more than 4

• If sorting 100 items is slow, sorting 400 items will be 16 times as slow!

• “Quadratic Growth” – generally regarded as bad
Better Sorting

• “quicksort”
• Number of steps for a problem of size $k$ is proportional to $k \log k$
  – Just a little more than $k$
  – Generally regarded as “good”
• In-class example
What about senators?

• We had to sort our data
  – A bigger problem would take a little more than proportionally more time for sorting
  – The “size” of the data is the number of senators times the number of votes!
• We had to build the “agree table”. For n senators and k votes, this involved multiplying two n x k matrices
• Time taken is proportional to $n^2k$
• Doing it for the House (where n is 4.3 times as big) will take about 16 times as long!
One of the recurrent ideas in computer science is *scale* and *scalability*

Will this thing I did for a small problem work well for a big one?

Wifi is great for your home...but one wifi access point for a city? No way!

We can analyze the matrix of distances between pairs of senators...

– What about distances between all possible pairs of webpages?
What have we learned so far

• High level
• Data
• Computational tools
High level

• The question/gather data/analyze/conclude/re-question cycle
• How to phrase a problem computationally
Data

• Types of data
  – XML
  – CSV
  – HTML
  – JSON (barely)

• How to look for data

• How to grab data from the web, using ImportXML or ImportHTML, or read a CSV file
Parameterized Web Queries

• If data is organized in a regular way...
• And the URL is formed in a regular way...
• We can turn “110th congress”, “session 1”, bill 215 into
  http://www.senate.gov/legislative/LIS/roll_call_votes/vote1102/vote_110_2_00215.xml
• A user of our spreadsheet needs only change “110” to “111” to get
  http://www.senate.gov/legislative/LIS/roll_call_votes/vote1112/vote_111_2_00215.xml
Parameterized Web Queries

• The user can enter a word to search, and we can extract how many webpages match
Computational Tools

- “If” expressions, nested “if”s
- Building complex expressions from simple ones
- Re-coding data (“Yea” -> 1, “Nay” -> -1)
- Using visualization to detect data anomalies
- Using formulas rather than copy/paste, so that spreadsheet is “live”: changing a parameter can update everything else
Computational Tools

• Pivot tables
  – For summarizing and reshaping data

• Matrix transpose and multiply
  – For repeated pairwise operations, after clever coding

• OFFSET, ROW, COLUMN
  – For structured copying or referencing of data

• Data-driven drop-down menus for parameters
  – To prevent bad data entry
Computational Tools

• String-handling
  – Concatenation with “&”
  – Breaking into pieces with “SPLIT”
  – Using MATCH
  – Not very convenient – better in Python

• Structuring of spreadsheets
  – Using multiple tabs for sequential steps
  – Labeling of tabs, cells
  – Use of comments/notes on cells for clarity
  – Color-coding cells for ease of reading/use
Set Up a Google Site in 5 Minutes
Set Up a Google Site in 5 Minutes

Hypothesis

My hypothesis is great.

Comments

Steven Gomez

Add a comment
Calendar

• Project 1 proposal is due tomorrow!
• Project 1 due in about 2 weeks
Project Goals

• Do something interesting to you (and perhaps to us, too!)
  – Your hypothesis doesn’t have to be correct
    • It’s nice if it’s wrong in an interesting way
• Do something that involves showing off your skills
  – Either substantial data analysis, or
  – Substantial work in data gathering, or
  – Substantial work in data assembly, or
  – A bit of each
• Expect to spend an honest 8 hours, and another hour on the website
Substantial Data Analysis

• The senate vote comparison
• The oddball senators remain the same from year to year
  – Find senator that’s least like any others
  – Or that is sorta like as few as possible
• Tweet topics/times of day are related
  – “Bieber”: afternoon to 2AM
  – “Sales”: 8AM to 5PM
Challenges in Data Gathering

• Load a list of top 100 books on Amazon...
  – And gather interesting information from the page for each one
• Figure out how to form a sequence of URLs effectively
• Determine the right Xpaths to get the data you want
Challenges in Data Assembly

• Gather data-by-country or data-by-state from multiple sources...
  – And do something to reconcile differing names
    • “Massachusetts” vs “Commonwealth of Massachusetts”

• Gather multi-year data...
  – And do something about missing values
Clustering

• Given a bunch of items and “distances” between them...
• Group them into “clusters”
• Things in one cluster should be closer to each other, on average, than to items in a different cluster
• There should be few clusters (e.g., not one per item!)
Clustering example
Clustering example
Clustering “algorithm”

• Pick a few “starting points” – say two
• For each data point, identify which starting-point is nearest
• Average together points in the same group, and use results as “starting points” for the next round
• Repeat until things look OK
Example
Example
Example
Example

Updated to average of orange points
Example
Example
Example
Example

Stable!
**k-means clustering**

- Stars are the “means”
- Tends to stabilize quickly when there really are clusters
- Tends to jump around when there are not
- You need to guess number of clusters *a priori*
- Data can lie in any dimension: just need to have a notion of “distance”
- Examples: Senators/votes; webpages and shared-words; geographic locations of tweets; ...