Lecture #2

Algorithms for Big Data

Joseph Bonneau
jcb82@cam.ac.uk
April 30, 2012
Today's topic: algorithms
Do we need new algorithms?

*Quantity is a quality of its own*

- Joseph Stalin, apocryphal

- can't always store all data
  - online/streaming algorithms
- memory vs. disk becomes critical
  - algorithms with limited passes
- $N^2$ is impossible
  - approximate algorithms
- human insight is limited
  - algorithms for high-dimensional data
Simple algorithms, more data

- *Mining of Massive Datasets*
  - Anand Rajaraman, Jeffrey Ullman 2010
  - Plus Stanford course, pieces adapted here
- “Synopsis Data Structures for Massive Data Sets”
  - Phillip Gibbons, Yossi Mattias, 1998
- “The Unreasonable Effectiveness of Data”
  - Alon Halevy, Peter Norvig, Fernando Perreira, 2010
The “easy” cases

- sorting
  - Google 1 trillion items, (1 PB) sorted in 6 hours
- searching
  - hashing & distributed search
Streaming algorithms

Have we seen $x$ before?
Bloom filter (Bloom 1970)

- assume we can store $n$ bits $b_1, b_2, \ldots, b_n$
- use a hash function $H(x) = h_1, h_2, \ldots, h_k$
  - each $h_i \in [1, n]$
- when we see $x$, set $b_i$, for each $i$ in $H(x)$

$n = 8$, $k = 4$

\[ B = \begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array} \]
Bloom filter (Bloom 1970)

- assume we can store $n$ bits $b_1, b_2, \ldots b_n$
- use a hash function $H(x) = h_1, h_2, \ldots h_k$
  - each $h_i \in [1, n]$
- when we see $x$, set $b_i$, for each $i$ in $H(x)$

$n = 8, k = 4$

$H(jcb82) = 4, 7, 5, 2$

$B = 0000 0000$
Bloom filter (Bloom 1970)

- assume we can store $n$ bits $b_1, b_2, \ldots b_n$
- use a hash function $H(x) = h_1, h_2, \ldots h_k$
  - each $h_i \in [1, n]$
- when we see $x$, set $b_i$, for each $i$ in $H(x)$

$n = 8, k = 4$  \hspace{2cm} $B = 0101 \ 1010$

$H(jcb82) = 4, 7, 5, 2$
$H(rja14) = 2, 4, 3, 8$
Bloom filter (Bloom 1970)

- assume we can store \( n \) bits \( b_1, b_2, \ldots, b_n \)
- use a hash function \( H(x) = h_1, h_2, \ldots, h_k \)
  - each \( h_i \in [1, n] \)
- when we see \( x \), set \( b_i \), for each \( i \) in \( H(x) \)

\[ n = 8, \quad k = 4 \]

\[
H(jcb82) = 4, 7, 5, 2 \\
H(rja14) = 2, 4, 3, 8 \\
H(mgk25) = 8, 3, 7, 1
\]

\[ B = 0111 \quad 1011 \]
Bloom filter (Bloom 1970)

- assume we can store $n$ bits $b_1, b_2, \ldots b_n$
- use a hash function $H(x) = h_1, h_2, \ldots h_k$
  - each $h_i \in [1, n]$
- when we see $x$, set $b_i$, for each $i$ in $H(x)$

$n = 8, k = 4 \quad B = 11111011$

$H(jcb82) = 4, 7, 5, 2$
$H(rja14) = 2, 4, 3, 8$
$H(mgk25) = 8, 3, 7, 1$
$H(fms27) = 1, 5, 2, 4$
Bloom filter (Bloom 1970)

- assume we can store \( n \) bits \( b_1, b_2, \ldots, b_n \)
- use a hash function \( \text{H}(x) = h_1, h_2, \ldots, h_k \)
  - each \( h_i \in [1, n] \)
- when we see \( x \), set \( b_i \), for each \( i \) in \( \text{H}(x) \)

\( n = 8, k = 4 \)

\[
\begin{align*}
\text{B} &= 1111 \ 1011 \\
\text{H}(jcb82) &= 4, 7, 5, 2 \\
\text{H}(rja14) &= 2, 4, 3, 8 \\
\text{H}(mgk25) &= 8, 3, 7, 1 \\
\text{H}(fms27) &= 1, 5, 2, 4
\end{align*}
\]

false positive
Bloom filters

- constant time lookup/insertion
- no false negatives
- false positives from $N$ items: $\ln p = \frac{-N/n}{(\ln 2)^2}$
  - at 1% $p$, 1 GB of RAM $\approx$ 77 billion unique items!
- once full, can't expand
- counting bloom filter: store integers instead of bits
Application: market baskets

- assume TESCO stocks 100k items
  - Amazon stocks many more
- market basket:
  - \{ravioli, pesto sauce, olive oil, wine\}
- perhaps 1 billion baskets per year
  - $2^{100,000}$ possible baskets...
- what items are “frequently” purchased together?
  - more frequent than predicted by chance
Application: market baskets

- pass #1: add all sub-baskets to counting BF
  - with 8 GB of RAM: 25B baskets
  - restrict basket size $\binom{100,000}{3} > 100$ trillion
- pass #2: check all sub-baskets
  - check $\{x\}, \{y_1, \ldots, y_n\} \{x, y_1, \ldots y_n\}$
  - store possible rules $\{y_1, \ldots, y_n\} \rightarrow \{x\}$
- pass #3: eliminate false positives
- for better approaches see Ramarajan/Ullman
Other streaming challenges

- rolling average of previous $k$ items
  - sliding window of traffic volume
- hot list – most frequent items seen so far
  - probabilistically start tracking new items
- histogram of values
Querying data streams

Select segNo, dir, hwy
From SegSpeedStr [Range 5 Minutes]
Group By segNo, dir, hwy
Having Avg(speed) < 40

- Continuous Query Language (CQL)
  - Oracle now shipping version 1
- based on SQL...
Similarity metrics

Which items are similar to $x$?
Similarity metrics

Which items are similar to $x$?

- reduce $x$ to a high-dimensional vector
- features:
  - words in a document (bag of words)
  - *shingles* (sequences of $w$ characters)
  - various body part measurements (faces)
  - edges, colours in a photograph
Feature extraction - images

Gist
Siagan, Itti, 2007
Distance metrics

- Chebyshev distance: $\max_i (x_i - y_i)$
- Manhattan distance: $\sum |x_i - y_i|$
- Hamming distance for binary vectors
- Euclidean distance: $\sqrt{\sum (x_i - y_i)^2}$
- Mahalanobis distance: $\sqrt{\sum \frac{(x_i - y_i)^2}{\sigma^2}}$
  - Adjusts for different variance
- Cosine distance: $\cos \theta = \frac{\sum x_i \cdot y_i}{|X| \cdot |Y|}$
- Jaccard distance (binary): $\frac{|X \cup Y| - |X \cap Y|}{|X \cup Y|}$
The curse of dimensionality

- $d = 2$: $\text{area}(\circ) = \pi$, $\text{area}(\square) = 4$
  - ratio = 0.78
- $d = 3$: $\text{area}(\circ) = (4/3) \pi$, $\text{area}(\square) = 8$
  - ratio = 0.52
- $d \rightarrow \infty$:
  - ratio $\rightarrow 0!$
- all points are “far” by Euclidean distance
The curse of dimensionality

- space is typically *very* sparse
- most dimensions are semantically useless
  - hard for humans to tell which ones
- need **dimension reduction**
Singular value decomposition

$$D_{[m \times n]} = U_{[m \times n]} \cdot \Sigma_{[n \times n]} \cdot V^T_{[n \times n]}$$

- books to words (input data)
- books to genres
- genre strength (diagonal)
- genres to words

see Baker '05
Singular value decomposition

books to words (input data)

books to words (approximate)

≈

≈

≈

≈

≈

books to genres

genre strength (diagonal)

genres to words

see Baker '05
**Singular value decomposition**

- **books to words (input data)**
  
  $ \begin{bmatrix}
  2 & 0 & 8 & 6 & 0 \\
  1 & 6 & 0 & 1 & 7 \\
  5 & 0 & 7 & 4 & 0 \\
  7 & 0 & 8 & 5 & 0 \\
  0 & 10 & 0 & 0 & 7 \\
  \end{bmatrix}$

- **books to words (approximate)**
  
  $ \approx \begin{bmatrix}
  2.29 & -0.66 & 9.33 & 1.25 & -3.09 \\
  1.77 & 6.76 & 0.90 & -5.50 & -2.13 \\
  4.86 & -0.96 & 8.01 & 0.38 & -0.97 \\
  6.62 & -1.23 & 9.58 & 0.24 & -0.71 \\
  1.14 & 9.19 & 0.33 & -7.19 & -3.13 \\
  \end{bmatrix}$

- **books to genres**
  
  $ \approx \begin{bmatrix}
  -0.54 & 0.07 & 0.82 \\
  -0.1 & -0.59 & -0.11 \\
  -0.53 & 0.06 & -0.21 \\
  -0.65 & 0.07 & -0.51 \\
  -0.06 & -0.8 & 0.09 \\
  \end{bmatrix}$

- **genre strength (diagonal)**
  
  $ \begin{bmatrix}
  17.92 & 0 & 0 \\
  0 & 15.17 & 0 \\
  0 & 0 & 3.56 \\
  \end{bmatrix}$

- **genres to words**
  
  $ \approx \begin{bmatrix}
  -0.46 & 0.02 & -0.87 & 0 & 0.17 \\
  -0.07 & -0.76 & 0.06 & 0.6 & 0.23 \\
  -0.74 & 0.1 & 0.28 & 0.22 & -0.56 \\
  \end{bmatrix}$

see Baker '05
Singular value decomposition

books to words (input data)

\[
\begin{bmatrix}
2 & 0 & 8 & 6 & 0 \\
1 & 6 & 0 & 1 & 7 \\
5 & 0 & 7 & 4 & 0 \\
7 & 0 & 8 & 5 & 0 \\
0 & 10 & 0 & 0 & 7
\end{bmatrix}
\]

≈

books to words (approximate)

\[
\begin{bmatrix}
2.29 & -0.66 & 9.33 & 1.25 & -3.09 \\
1.77 & 6.76 & 0.90 & -5.50 & -2.13 \\
4.86 & -0.96 & 8.01 & 0.38 & -0.97 \\
6.62 & -1.23 & 9.58 & 0.24 & -0.71 \\
1.14 & 9.19 & 0.33 & -7.19 & -3.13
\end{bmatrix}
\]

≈

genres to words

\[
\begin{bmatrix}
17.92 & 0 & 0 \\
0 & 15.17 & 0 \\
0 & 0 & 3.56
\end{bmatrix}
\]

genre strength (diagonal)

\[
\begin{bmatrix}
-0.46 & 0.02 & -0.87 & 0 & 0.17 \\
-0.07 & -0.76 & 0.06 & 0.6 & 0.23 \\
-0.74 & 0.1 & 0.28 & 0.22 & -0.56
\end{bmatrix}
\]

books to genres

genres to words

see Baker '05
Singular value decomposition

- computation takes $O(mn^2)$, with $m > n$
- useful but out-of-reach for largest datasets
- implemented in most statistics packages
  - R, MATLAB, etc
- (often) better linear algebra approaches exist
  - CUR, CMD decomposition
Locality-sensitive hashing

- replace SVD with simple probabilistic approach
- choose a family of hash functions $H$ such that:
  - $\text{distance}(X, Y) \approx \text{distance}(H(X), H(Y))$
  - $H$ can produce any number of bits
- compute several different $H$
- investigate further if $H(X) = H(Y)$ exactly
  - scale output size of $H$ to minimise cost
MinHash implementation

- compute a random permutation $\sigma_X(R)$
- count the number of 0's before a 1 appears
- **theorem:**
  - $\text{pr}[\text{MH}(X) = \text{MH}(Y)] = 1 - \text{Jaccard}(X, Y)$
- combine multiple permutations to add bits
LSH example - Flickr photos

Kulis and Grauman, 2010
Can we recommend books, films, products to users based on their personal tastes?

Recommended for you

**The Aviator** (2004)
PG-13 Biography | Drama

7.5/10

A biopic depicting the early years of legendary director and aviator Howard Hughes' career, from the late 1920s to the mid-1940s.

**Director:** Martin Scorsese

**Stars:** Leonardo DiCaprio and Cate ...
Recommendation systems

ANATOMY OF THE LONG TAIL
Online services carry far more inventory than traditional retailers. Rhapsody, for example, offers 19 times as many songs as Wal-Mart’s stock of 39,000 tunes. The appetite for Rhapsody’s more obscure tunes (charted below in yellow) makes up the so-called Long Tail. Meanwhile, even as consumers flock to mainstream books, music, and films (right), there is real demand for niche fare found only online.

THE NEW GROWTH MARKET: OBSCURE PRODUCTS YOU CAN’T GET ANYWHERE BUT ONLINE

Sources: Erik Brynjolfsson and Jeffrey Hu, MIT, and Michael Smith, Carnegie Mellon; Barnes & Noble; Netflix; RealNetworks

Anderson 2004
Content-based filtering

- extract features from content
  - actors
  - director
  - genre
  - keywords in plot summary
  - etc.

- find nearest-neighbours to what a user likes or buys
Content-based filtering

- **PROS**
  - rate new items
  - no herding
  - no user information exposed to others

- **CONS**
  - features may not be relevant
  - recommendations may be boring
    - “filter bubble”
Collaborative filtering

- features are user/item interactions
  - purchases
  - explicit ratings
    - need lots of clean-up, scaling
- user-user filtering: find similar users
  - suggest their top ratings
  - scale for each user's rating style
- item-item filtering: find similar items
  - suggest most similar items
Item-item filtering preferred

Koren, Bell, Volinksy, 2009
Collaborative filtering

- **PROS**
  - automatic feature extraction
  - surprising recommendations

- **CONS**
  - ratings for new users/items
  - herding
  - privacy
The Netflix Prize, 2006-2009

- 100M film ratings made available
  - 480k users
  - 17k films
  - (shoddily) anonymised
- 3M ratings held in reserve
  - goal: improve predictions by 10%
  - measure by RMS error
- US $1 M prize
  - attracted over 2500 teams
Netflix Prize insights

- need to heavily normalise user ratings
  - some users more critical than others
  - user rating style
  - temporal bias
- latent item factors is strongest approach
# Netflix Prize league table

<table>
<thead>
<tr>
<th>Rank</th>
<th>Team Name</th>
<th>Best Test Score</th>
<th>% Improvement</th>
<th>Best Submit Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BellKor's Pragmatic Chaos</td>
<td>0.8567</td>
<td>10.06</td>
<td>2009-07-26 18:18:28</td>
</tr>
<tr>
<td>2</td>
<td>The Ensemble</td>
<td>0.8567</td>
<td>10.06</td>
<td>2009-07-26 18:38:22</td>
</tr>
<tr>
<td>3</td>
<td>Grand Prize Team</td>
<td>0.8582</td>
<td>9.90</td>
<td>2009-07-10 21:24:40</td>
</tr>
<tr>
<td>4</td>
<td>Opera Solutions and Vandelay United</td>
<td>0.8588</td>
<td>9.84</td>
<td>2009-07-10 01:12:31</td>
</tr>
<tr>
<td>5</td>
<td>Vandelay Industries</td>
<td>0.8591</td>
<td>9.81</td>
<td>2009-07-10 00:32:20</td>
</tr>
<tr>
<td>6</td>
<td>PragmaticTheory</td>
<td>0.8594</td>
<td>9.77</td>
<td>2009-06-24 12:06:25</td>
</tr>
<tr>
<td>7</td>
<td>BellKor in BigChaos</td>
<td>0.8601</td>
<td>9.70</td>
<td>2009-05-13 08:14:09</td>
</tr>
<tr>
<td>8</td>
<td>Dace</td>
<td>0.8612</td>
<td>9.59</td>
<td>2009-07-24 17:18:43</td>
</tr>
<tr>
<td>9</td>
<td>Feeds2</td>
<td>0.8622</td>
<td>9.48</td>
<td>2009-07-12 13:11:51</td>
</tr>
<tr>
<td>10</td>
<td>BigChaos</td>
<td>0.8623</td>
<td>9.47</td>
<td>2009-04-07 12:33:59</td>
</tr>
<tr>
<td>11</td>
<td>Opera Solutions</td>
<td>0.8623</td>
<td>9.47</td>
<td>2009-07-24 09:34:07</td>
</tr>
<tr>
<td>12</td>
<td>BellKor</td>
<td>0.8624</td>
<td>9.46</td>
<td>2009-07-26 17:19:11</td>
</tr>
</tbody>
</table>
Netflix Prize aftermath

- winning solution is a messy hybrid
- never implemented in practice!
  - too much pain for little gain...
Can we identify *groups* of similar items?
Can we identify *groups* of similar items?
Clustering

- identify categories of:
  - organisms (species)
  - customers (tastes)
  - graph nodes (community detection)
- develop cluster scores based on item similarity
  - diameter of clusters?
  - radius of clusters?
  - average distance?
  - distance from other clusters?
Hierarchical clustering

Lindblad-Toh et al. 2005

O(N^2 \log N)
Approximate $k$-means

- choose # of categories $k$ in advance
  - can test various values
- draw a random “RAM-full” of items
- cluster them “optimally” into $k$ clusters
- identify the “centre”
  - centroid: average of points (Euclidean)
  - clustroid: closest to other points
- assign remaining points to closest centre
  - $O(N)$ time
Given some examples, can we classify new items?
Classification

- is this item...
  - a spam email?
  - a cancerous cell?
  - the letter 'J'?
- many approaches exist:
  - neural networks
  - Bayesian decision trees
  - domain-specific probabilistic models
Manual classification

What sentiments does this image convey?

Please choose at least one sentiment:
- active
- euphoric
- funny
- happy
- calm/comforting
- miserable
- nothing
- passive
- overall impression

What triggered that sentiment the most?

mostly overall impression

motif
Manual classification

What do you see?

taboo words
grandmother
outside

Partner clicked pass
Play Anonymously

score 0
time 2:27

guesses

ESP Game, Von Ahn et al.
**k-nearest neighbour classifier**

Classify a point by majority vote of its $k$ nearest neighbors
$k$-nearest neighbour classifier

$k$ can make a big difference!
Bias-variance tradeoff

too much variance
Bias-variance tradeoff

too much bias
In high-dimensional space, linear is often best
(If not? Map to a different space and linearize-\textit{kernel machines})
Support vector machines (SVM)

find hyperplane with maximal distance between any points
- closest points define the plane (support vectors)
**SVM example - cat or dog?**

<table>
<thead>
<tr>
<th>Cats</th>
<th>Dogs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most dog-like</td>
<td>Most cat-like</td>
</tr>
<tr>
<td>![Cat image]</td>
<td>![Dog image]</td>
</tr>
<tr>
<td>![Cat image]</td>
<td>![Dog image]</td>
</tr>
</tbody>
</table>

Golle 2007
Machine learning for dummies

- many other algorithms for classifying/clustering
  - learn the concepts and what you can tweak
- let others do the hard work
  - libraries: Shogun, libsvm, scikit-learn
  - Apache Mahout: works with Hadoop!
- outsource to Kaggle...
  - more in the next lecture
Thank you