Think Small

B. W. Lewis
6/10/2015
Think about solving your problems before trying to implement them
Think about solving your problems before trying to implement them

*especially on distributed systems*
Sometimes, evidently huge problems are **smaller** than they seem.
Outline

- Examples
- Practical tools
Big data...it's a nonsense term.

—Bill Cleveland, Interface 2015
Our conclusion is not that the systems ... are obviously bad, but rather that there is not yet much evidence that they are especially good.

—Frank McSherry on Spark, GraphX, Giraph, GraphLab
Page rank
Twitter_rv

<table>
<thead>
<tr>
<th>System</th>
<th>CPU Cores</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spark/GraphX</td>
<td>128</td>
<td>419</td>
</tr>
<tr>
<td>Giraph</td>
<td>128</td>
<td>596</td>
</tr>
<tr>
<td>Graphlab</td>
<td>128</td>
<td>249</td>
</tr>
<tr>
<td>McSherry’s Macbook Pro</td>
<td>1</td>
<td>110</td>
</tr>
</tbody>
</table>

Follower network example, ~41M vertices 1.4B edges

This table shows 20 iterations of page rank computed by Frank McSherry on his laptop, compared to some big distributed systems.
Sparse SVD biclustering
Innovative algorithm by Lee, Shen, Huang, and Marron

and a faithful R package implementation (s4vd) by Martin Sill and Sebastian Kaiser
Findings

- The method requires only a low-rank SVD in each iteration
- An inner matrix optimization problem could be (exactly) projected into a 1-d problem

These knock an $m^3$ problem down to about an order $m$ problem or so, where $m$ is the number of data observations
Example

1,000 x 10,000 example matrix timed on a Macbook Pro laptop

```r
library(s4vd)
set.seed(1)
x = matrix(rnorm(1000 * 10000), nrow=1000)
i = seq(from=1, to=nrow(x), by=2)
j = seq(from=1, to=ncol(x), by=2)
x[i,j] = rnorm(length(i)*length(j), mean=2, sd=0.5)
cl = biclust(x, method=BCssvd, K=1)
```

<table>
<thead>
<tr>
<th>Original routines</th>
<th>3,522s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reformulated routines</td>
<td>20s</td>
</tr>
</tbody>
</table>
Network centrality
A tiny part of the Bitcoin transaction graph

(If you don't see a plot, press reload.)
Viewed as an adjacency matrix $A$:
Network centrality

It's easy to see that

$$(A^k)_{i,j}$$

counts the number of paths of length $k$ between vertices $i$ and $j$. 
Network centrality

\[
(A + A^2 + A^3 + \cdots)_{i,j}
\]

counts the number of paths of all lengths between vertices \(i\) and \(j\).
Network centrality

One interesting measure of centrality de-emphasizes longer paths

\[ I + 1A + \frac{1}{2!}A^2 + \cdots = \exp(A). \]

Can be expensive to compute.
Network centrality

Rodriguez et al. show that the top $k$ most central network nodes are often found in low-dimensional subspaces related to the SVD.

And, they have a simple method that tells us when we've got a subspace guaranteed to contain the top $k$ most central nodes.
Bitcoin network centrality on a Chromebook

load("bitcoin_from_to_graph.rdata")
t1 = proc.time()
x = topm(8,g=2,tol=0.1,m_b=5)
proc.time() - t1

# user system elapsed
# 86.970 24.350 111.605

Compute the top 5 most central nodes for the *entire* Bitcoin transaction graph.
Compare to Padé approximant

On a 1000 X 1000 subset

```r
t1 = proc.time()
ex = diag(expm(X) + expm(-X))/2
proc.time() - t1
# user system elapsed
# 151.680 0.220 151.552

order(ex,decreasing=TRUE)[1:5]
#[1] 11 25 27 29 74
```

```r
t1 = proc.time()
top = topm(X, type="cent")
proc.time() - t1
# user system elapsed
# 0.555 0.010 0.565

top$hubs
#[1] 11 25 27 29 74
```
Google genomics and Spark (ADAM): not thinking small enough
PCA on the 1000 genomes chrom. 20

Often used as an example of a big data analysis...

I'm not saying this is profound or even interesting. Just that it's easy. A few minutes from start to finish on a laptop.
PCA on the 1000 genomes chrom. 20

```r
library(Matrix)
p = pipe("zcat ALL.chr20.phase3....genotypes.vcf.gz
     | sed /^#/d | cut -f '10-' | parser | cut -f '1-2'")
x = read.table(p, colClasses=c("integer","integer"), fill=TRUE)
chr20 = sparseMatrix(i=x[,2], j=x[,1], x=1.0)

print(dim(chr20))
# [1] 2504 1812841

library(irllba)
cm = colMeans(chr20)
p = irllba(chr20, nv=3, nu=3, du=rep(1,nrow(chr20)), ds=1, dv=cm)

library(threejs)
scatterplot3js(p$u)
```
What about more chromosomes?

Not too big a deal. Easy to supply a custom, column-wise chunked matrix-vector product.
Linear mixed models
FastLMM

Nifty recent work of Lippert, Listgarten, Liu, Kadie, Davidson, Heckerman at MSFT Research

*Beautifully simple* change of basis that reduces normally expensive large-scale mixed effects models to simpler (diagonal) form

Can be combined naturally with low-dimensional subspace projection
Practical tools
Projection methods turn “big data” into data
Projection methods turn “big data” into data

Changing bases can exploit underlying structure to simplify problems
IRLBA

- State of the art partial SVD by Baglama and Reichel
- Competitive with Martinsson, Rokhlin, et al.'s methods.
- On CRAN; recently extensions to network centrality, partial symmetric eigensystems
References

- http://www.frankmcsherry.org/graph/scalability/cost/2015/01/15/COST.html
- https://github.com/mwills/s4vd (sparse svd biclustering)
- http://compbio.cs.uic.edu/data/bitcoin/ (bitcoin data)
- https://github.com/googlegenomics/spark-examples
- http://htmlwidgets.org (the interactive R plots)
- https://github.com/bwlewis/IRL (iriba method, also on CRAN)
- https://github.com/bwlewis/1000_genomes_examples
- http://www.nature.com/nmeth/journal/v8/n10/abs/nmeth.1681.html (Fast LMM)
- http://illposed.net