

CSCI 6900: Mining Massive Datasets

Shannon Quinn

(with content graciously and viciously borrowed from William Cohen's 10-605 Machine Learning with Big Data and Stanford's MMDS MOOC <http://www.mmids.org/>)

“Big Data”

SAY BIG DATA

ONE MORE TIME

memegenerator.net



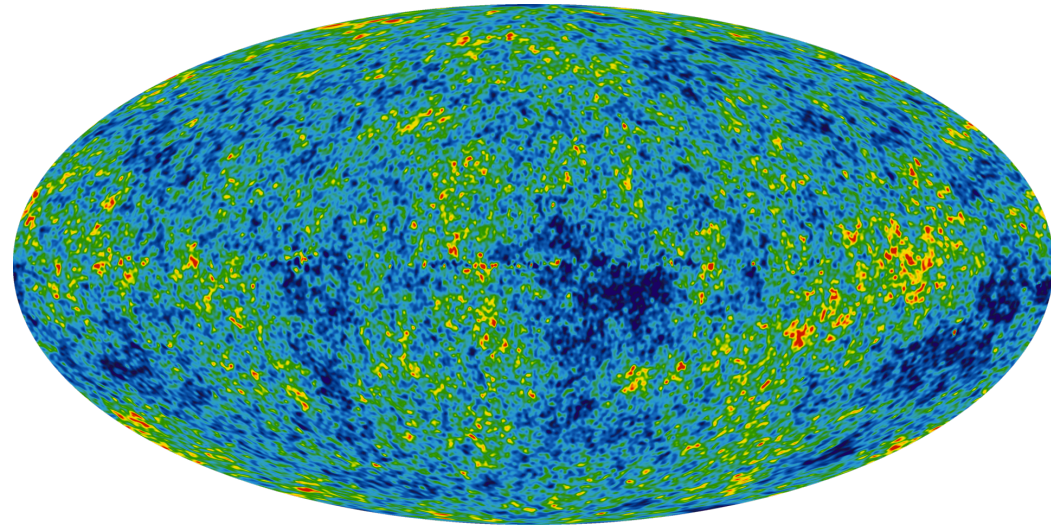
BIG DATA

**IT DOES NOT MEAN WHAT YOU
THINK IT MEANS**



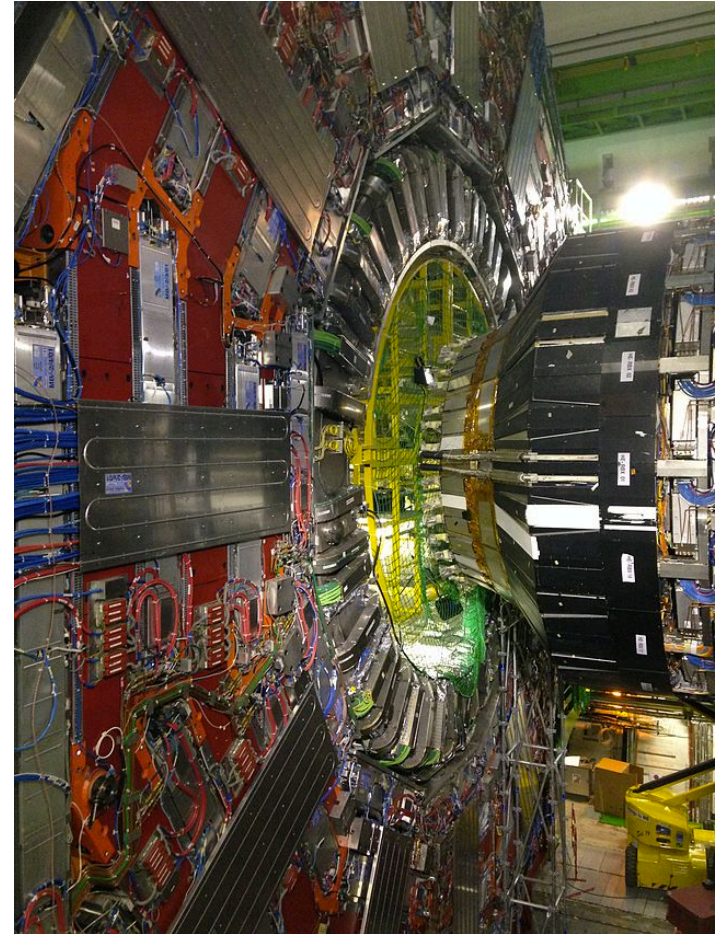
Astronomy

- Sloan Digital Sky Survey
 - New Mexico, 2000
 - 140TB over 10 years
- Large Synoptic Survey Telescope
 - Chile, 2016
 - Will acquire 140TB **every five days**¹



Particle Physics

- Large Hadron Collider (LHC)
 - 150 million sensors
 - 40 million data points / second (before filtering)
 - 100 collisions of interest (after filtering)¹
 - Even after rejecting 199,999 of every 200,000 collisions, generates **15PB** of data per year^{1,2}
 - If all collisions were recorded, LHC would generate 500EB of data **per day**
 - ~900EB transmitted over IP **per year**³



¹ <http://cds.cern.ch/record/1092437/files/CERN-Brochure-2008-001-Eng.pdf>

² <http://www.nature.com/news/2011/110119/full/469282a.html>

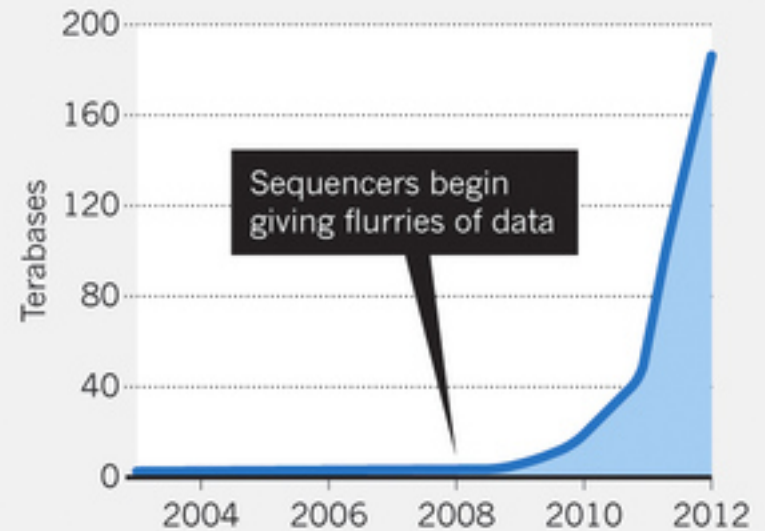
³ http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/VNI_Hyperconnectivity_WP.html

Biology

- Nucleotide sequences from **120,000+ species** in GenBank¹
- European Bioinformatics Institute (EBI)
 - **20PB of data** (genomic data doubles in size **each year**)²
 - A single sequenced human genome can be around **140GB in size**²
- Heterogeneous data, spread out over many labs

DATA EXPLOSION

The amount of genetic sequencing data stored at the European Bioinformatics Institute takes less than a year to double in size.



¹ <http://www.nature.com/nature/journal/v455/n7209/full/455047a.html>

² <http://www.nature.com/nature/journal/v498/n7453/full/498255a.html>

\$600 to buy a disk drive that can
store all of the world's music

5 billion mobile phones
in use in 2010

30 billion pieces of content shared
on Facebook every month

40% projected growth in
global data generated
per year vs.

5%
growth in global
IT spending

\$5 million vs. \$400

Price of the fastest supercomputer in 1975¹
and an iPhone 4 with equal performance

235 terabytes data collected by
the US Library of Congress
by April 2011

15 out of 17
sectors in the United States have
more data stored per company
than the US Library of Congress

Data Mining

- Knowledge discovery
 - “Big Data”
 - “Predictive Analysis”
 - “Data Science”

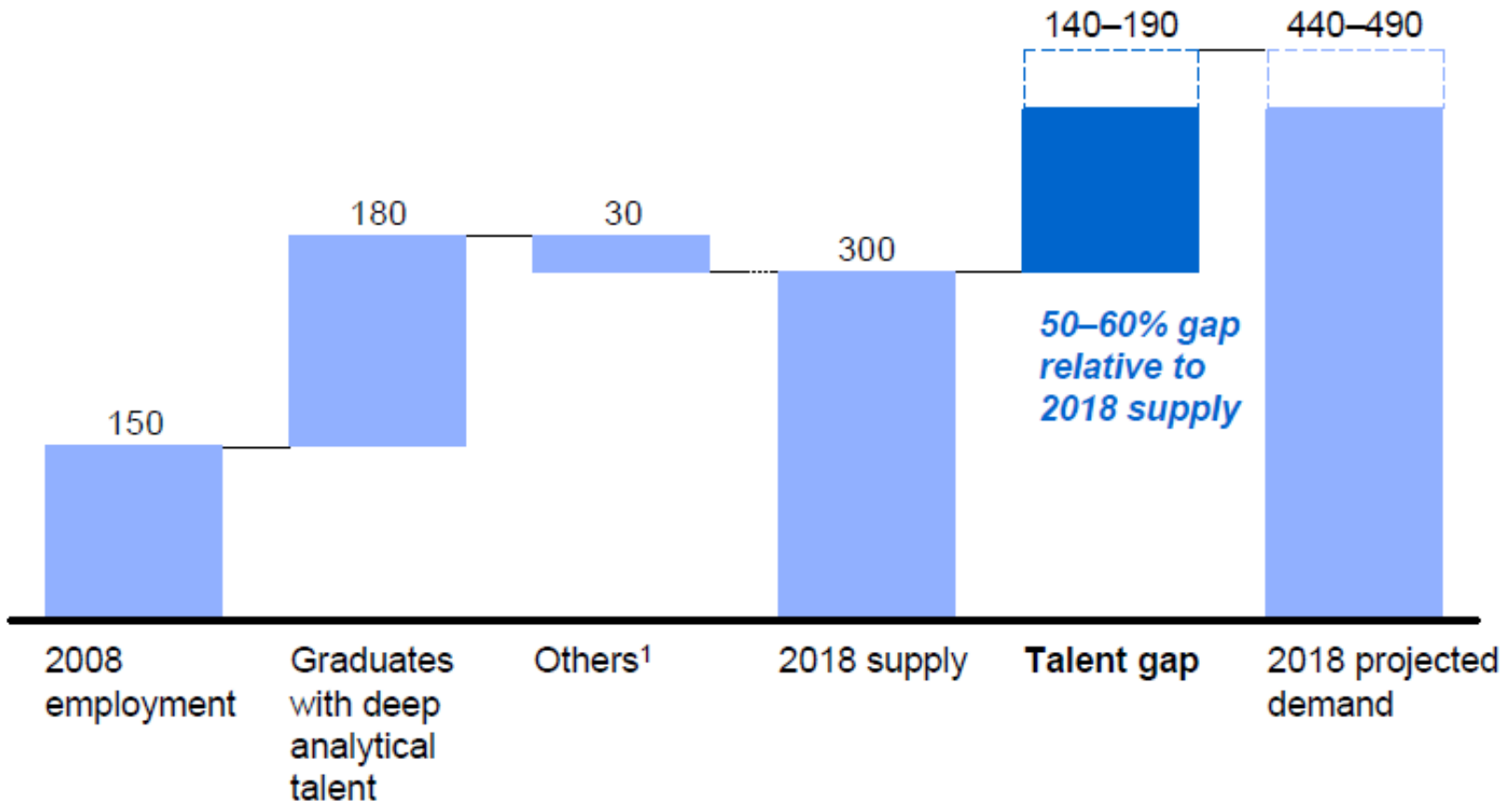


Data Scientists in demand

Demand for deep analytical talent in the United States could be 50 to 60 percent greater than its projected supply by 2018

Supply and demand of deep analytical talent by 2018

Thousand people



¹ Other supply drivers include attrition (-), immigration (+), and reemploying previously unemployed deep analytical talent (+).

Why is large-scale data mining a thing?

- Why not use the same algorithms on larger data?

Big ML c. 1993 (Cohen, “Efficient...Rule Learning”, IJCAI 1993)

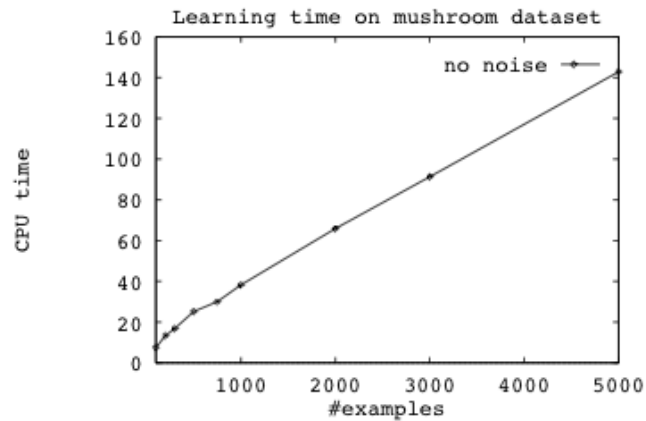


Figure 1: Rule induction without noise

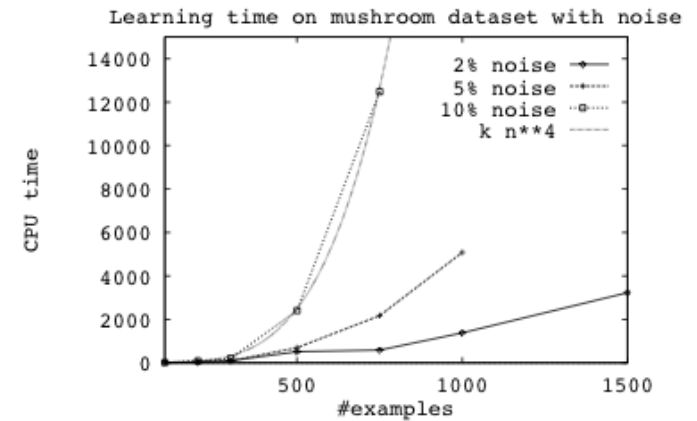
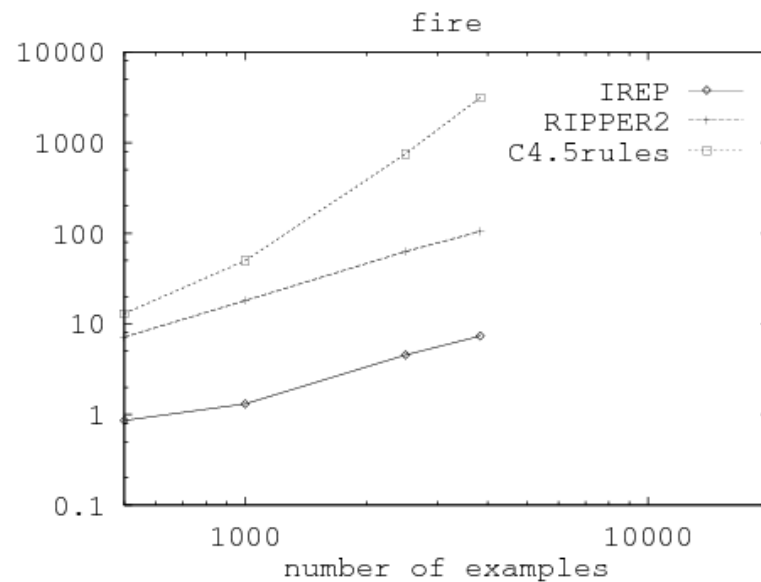
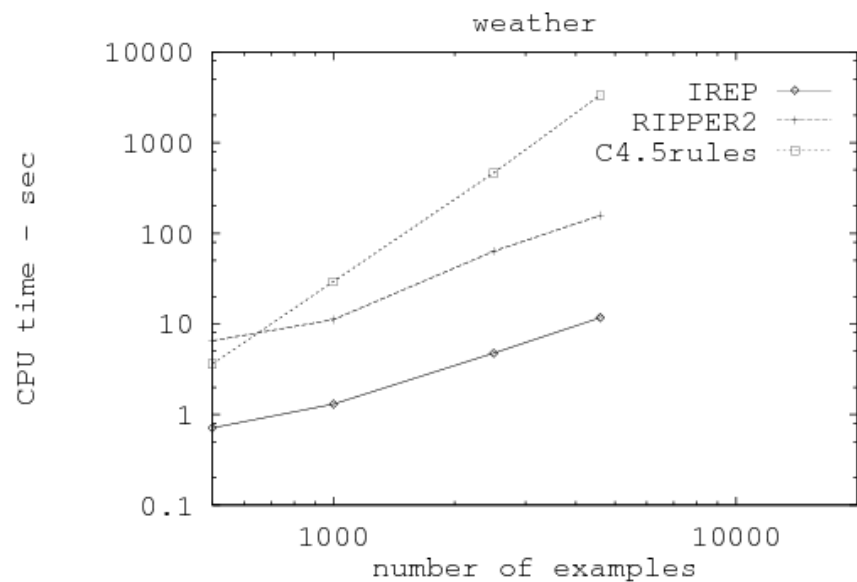
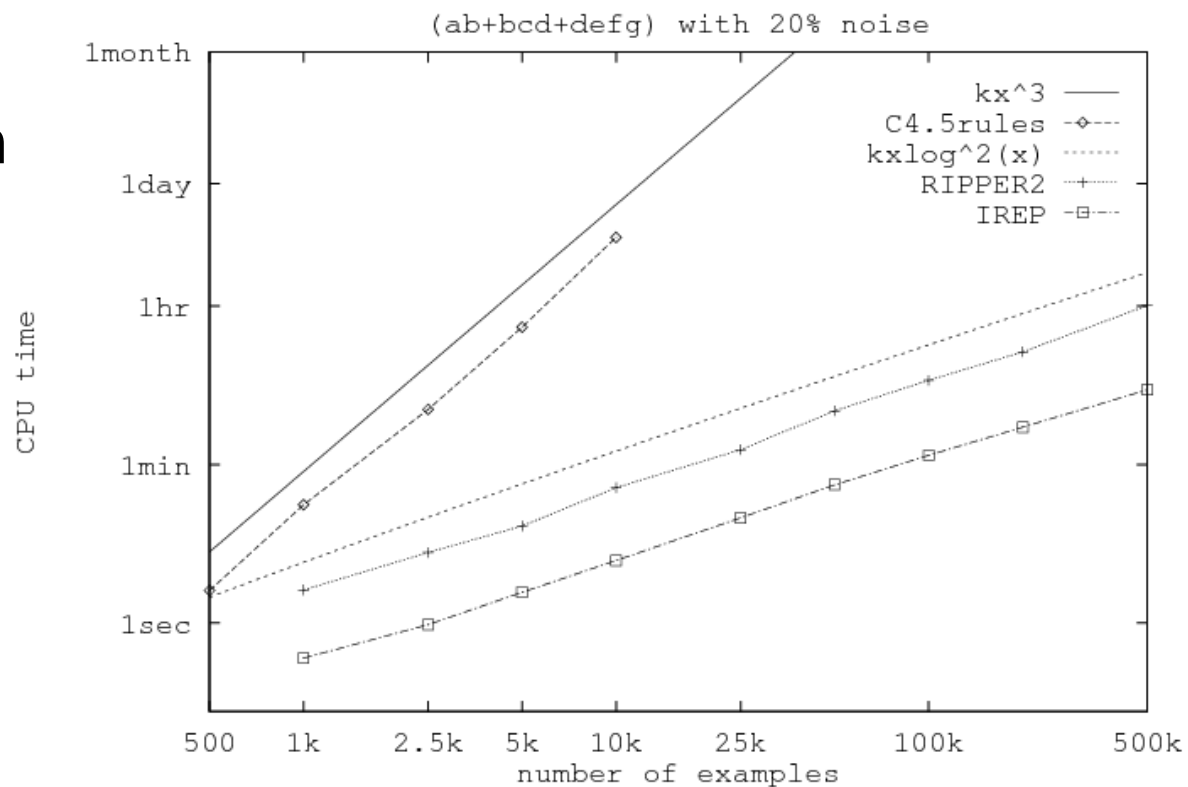


Figure 2: Rule induction with noise

Benchmark	CPU Time							
	No Pruning		REP		Grow		MDLGrow	
kr-vs-kkn	10.8	± 0.6	18.5	± 1.8	13.2	± 0.6	13.4	± 0.6
bridge-t/d	12.7	0.9	27.6	3.3	10.4	0.9	8.1	0.7
thyroid-hypo	72.6	6.5	56.6	9.4	46.4	6.2	48.1	6.3
bridge-mtrl	22.1	0.6	76.7	9.6	16.5	1.5	10.6	0.6
mushroom	35.6	0.8	78.3	7.8	44.5	1.4	45.3	1.7
thyroid-allbp	144.8	7.7	164.5	12.6	99.5	4.6	100.7	5.8
bridge-span	29.5	0.8	176.2	18.6	31.9	2.3	13.3	0.8
bridge-rel-l	44.1	1.0	294.1	36.9	34.0	2.9	14.1	1.2
bridge-type	38.9	1.1	370.6	25.2	40.5	2.3	21.2	1.0
sonar	561.0	12.9	399.2	15.1	368.2	12.0	370.6	12.1
segment	815.7	23.9	1264.0	86.6	728.2	29.6	733.6	27.8
mushroom*	217.2	10.1	4081.7	485.4	276.7	23.4	135.1	6.9
kr-vs-kkn*	154.2	11.9	5549.3	1255.3	206.6	23.5	53.5	4.0
rds	3189.1	84.9	15155.2	1282.4	2210.0	52.0	879.9	42.4
Average for Benchmark Set 2	382.03		2695.63		402.00		239.38	
Average for Benchmark Set 1	108.4		384.0		105.9		100.5	

Table 3: Comparing runtimes

Related paper from 1995...



So in mid 1990's.....

- Experimental datasets were small
- Many commonly used algorithms were *asymptotically* “slow”

Big ML c. 2001 (Banko & Brill, “Scaling to Very Very Large...”, ACL 2001)

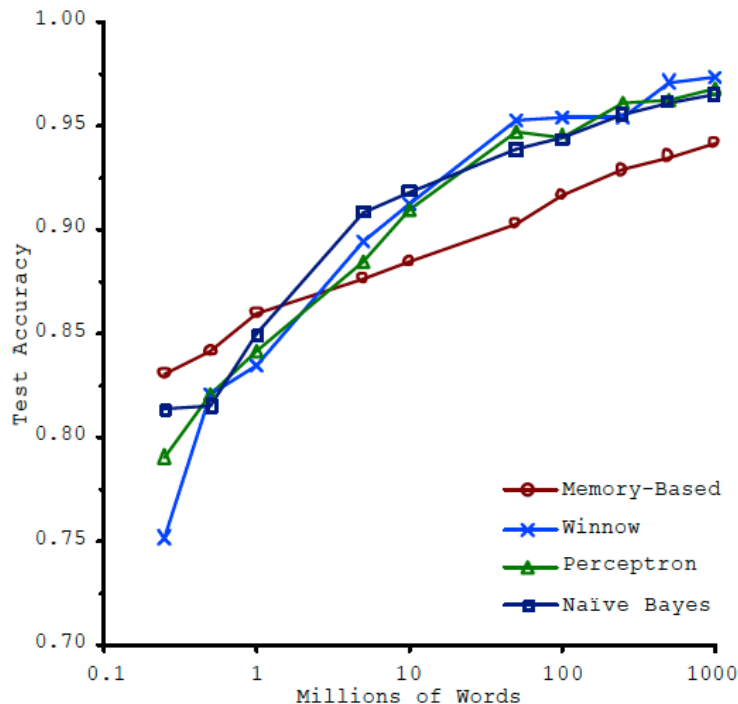


Figure 1. Learning Curves for Confusion Set Disambiguation

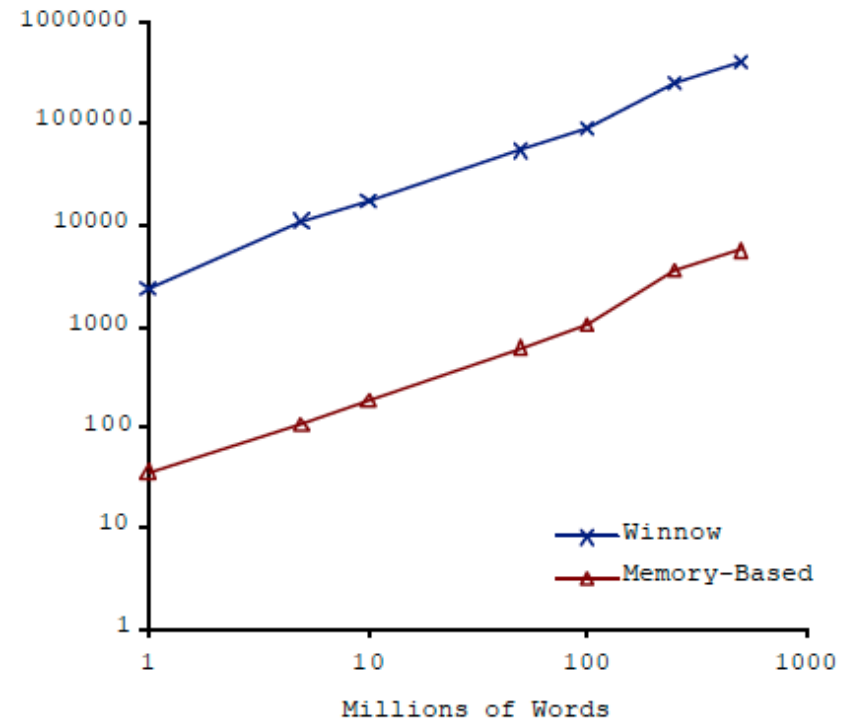
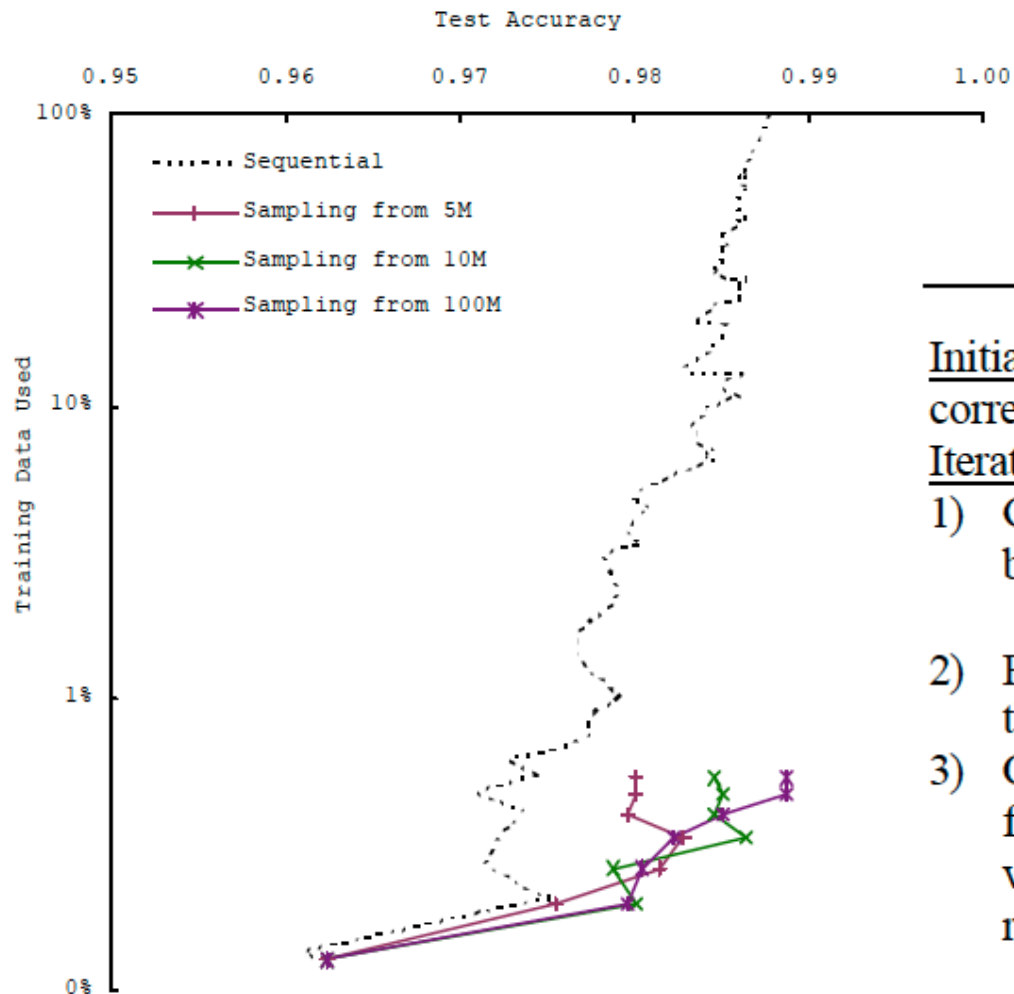


Figure 2. Representation Size vs. Training Corpus Size

Task: distinguish pairs of easily-confused words (“affect” vs “effect”) in context

Big ML c. 2001 (Banko & Brill, “Scaling to Very Very Large...”, ACL 2001)



Initialize: Training data consists of X words correctly labeled

Iterate:

- 1) Generate a committee of classifiers using bagging on the training set
 - 2) Run the committee on unlabeled portion of the training set
 - 3) Choose M instances from the unlabeled set for labeling - pick the M/2 with the greatest vote entropy and then pick another M/2 randomly – and add to training set
-

Figure 4. Active Learning with Large Corpora

So in 2001.....

- We're learning:
 - “there's no data like more data”
 - For many tasks, there's no real *substitute* for using lots of data

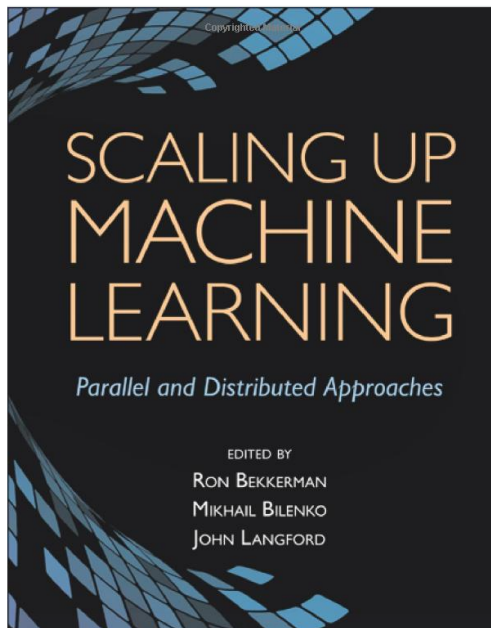
...and in 2009

Eugene Wigner's article "*The Unreasonable Effectiveness of Mathematics in the Natural Sciences*" examines why so much of physics can be neatly explained with simple mathematical formulas such as $f = ma$ or $e = mc^2$. Meanwhile, sciences that involve human beings rather than elementary particles have proven more resistant to elegant mathematics. Economists suffer from physics envy over their inability to neatly model human behavior. An informal, incomplete grammar of the English language runs over 1,700 pages.

Perhaps when it comes to natural language processing and related fields, we're doomed to complex theories that will never have the elegance of physics equations. But if that's so, we should stop acting as if our goal is to author extremely elegant theories, and instead embrace complexity and make use of the best ally we have: the unreasonable effectiveness of data.

Norvig, Pereira, Halevy, "The Unreasonable Effectiveness of Data", 2009

...and in 2012



Arthur Gretton, Michael Mahoney, Mehryar Mohri, Ameet Talwalkar

Gatsby Unit, UCL; Stanford; Google Research; UC Berkeley

Workshop: Low-rank Methods for Large-scale Machine Learning

7:30am - 6:30pm Saturday, December 11, 2010

Joseph Gonzalez, Sameer Singh, Graham Taylor, James Bergstra, Alice Zheng, Misha Bilenko, Yucheng Low, Yoshua Bengio, Michael Franklin, Carlos Guestrin, Andrew McCallum, Alexander Smola, Michael Jordan, Sugato Basu

Carnegie Mellon University; University of Massachusetts, Amherst; New York University; Harvard; Microsoft Research; Microsoft Research; Carnegie Mellon University; University of Montreal; UC Berkeley; Carnegie Mellon University; UMass Amherst; Yahoo! Research; University of California; Google Research

Workshop: Big Learning: Algorithms, Systems, and Tools for Learning at Scale

Location: Montebajo Theater

Dec 2011

SMLA Workshop 2010

29 June - 01 July, 2010, Bradford, UK

International Workshop on
Scalable Machine Learning and Applications (SMLA-10)
In conjunction with CIT 2010

...and in 2013

news ▶ opinion ▶ commentary

Forget YOLO: Why 'Big Data' Should Be The Word Of The Year

by GEOFF NUNBERG

December 20, 2012 10:58 AM

Listen to the Story 

+ Playlist

One of the biggest emerging stories about the campaign that has ended is how Mr. Obama's team used information and technology to outmatch and outwit a galvanized and incredibly well-financed opposition.



Adam Gryko/iStock

probably "frankens it was the buzz of Silicon Valley" like *Wired* and *The Economist*, and it was the buzz of Silicon Valley and Davos. And if the phrase wasn't as familiar to many people as "Etch A Sketch" and "47 percent," Big Data had just as much to do with President Obama's victory as they did.

Whether it's explicitly mentioned or not, the Big Data phenomenon has been all about intrusions on our data sweeps or the ads that track us as we wander around the Web. It has even turned statistics into a sexy major. So if you haven't heard the phrase yet, there's still time — it will be around a lot longer than "gangnam style."

How do we use very large amounts of data?

- Working with big data is ^{*}*not* about
 - code optimization
 - learning details of today's hardware/software:
 - GraphLab, Hadoop, parallel hardware,
- Working with big data *is* about
 - Understanding the cost of what you want to do
 - Understanding what the tools that are available offer
 - Understanding how much can be accomplished with linear or nearly-linear operations (e.g., sorting, ...)
 - Understanding how to organize your computations so that they effectively use whatever's fast
 - Understanding how to test/debug/verify with large data

* according to William Cohen / Shannon Quinn

Asymptotic Analysis: Basic Principles

Usually we only care about positive $f(n)$, $g(n)$, n here...

$$f(n) \in O(g(n)) \text{ iff } \exists k, n_0 : \forall n > n_0, f(n) \leq k \cdot g(n)$$

$$f(n) \in \Omega(g(n)) \text{ iff } \exists k, n_0 : \forall n > n_0, f(n) \geq k \cdot g(n)$$

Asymptotic Analysis: Basic Principles

Less pedantically:

$$f(n) = O(g(n)) \text{ iff } \exists k, n_0 : \forall n > n_0, f(n) \leq k \cdot g(n)$$

$$f(n) = \Omega(g(n)) \text{ iff } \exists k, n_0 : \forall n > n_0, f(n) \geq k \cdot g(n)$$

Some useful rules:

$$O(n^4 + n^3) = O(n^4)$$

Only highest-order terms matter

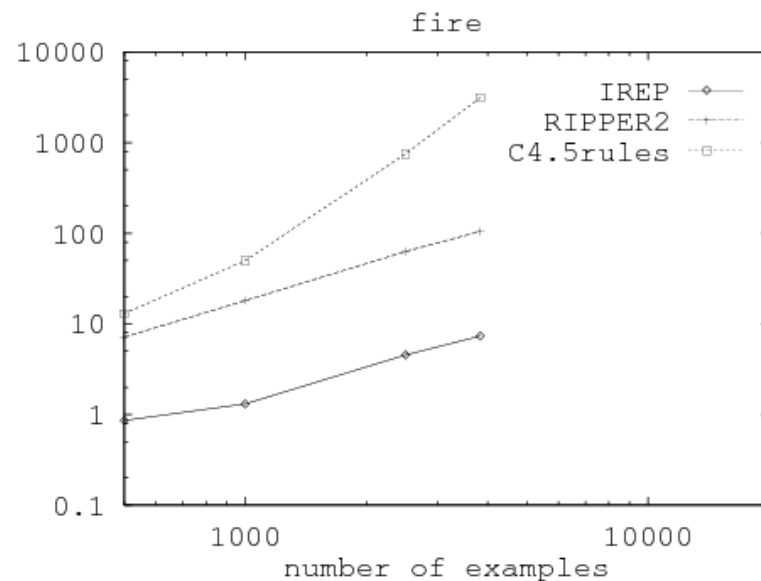
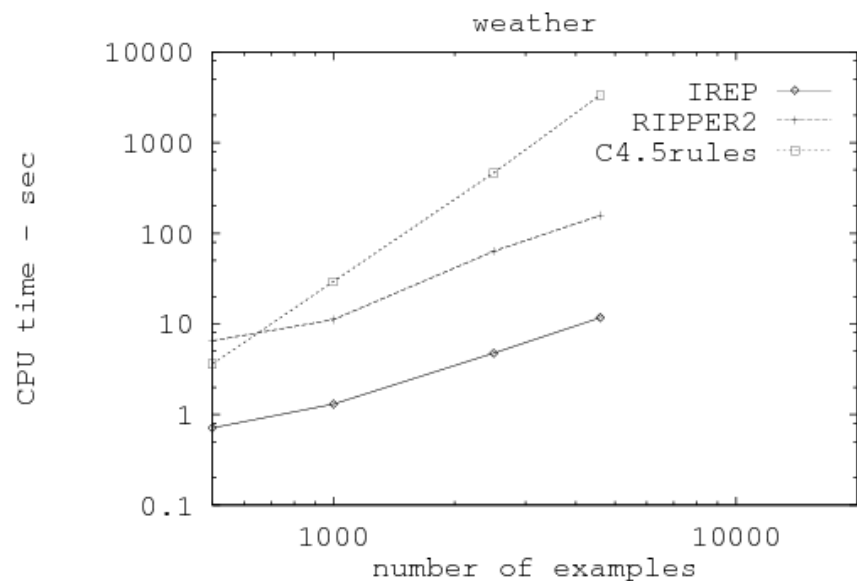
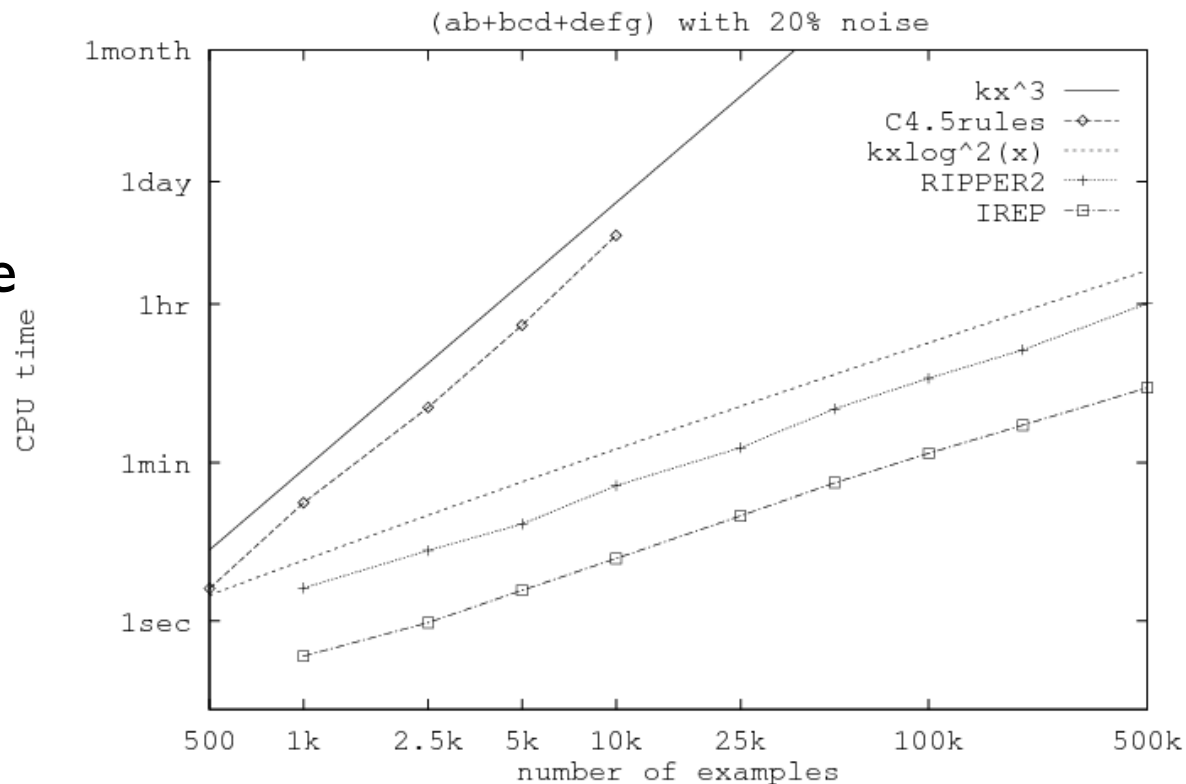
$$O(3n^4 + 127n^3) = O(n^4)$$

Leading constants don't matter

$$O(\log n^4) = O(4 \cdot \log n) = O(\log n)$$

Degree of something in a log doesn't matter

Empirical
analysis of
complexity:
plot run-time
on a log-log
plot and
measure the
slope (using
linear
regression)



Where do asymptotics break down?

- When the constants are too big
 - or n is too small
- When we can't predict what the program will do
 - Eg, how many iterations before convergence?
Does it depend on data size or not?
- When there are different types of operations with different costs
 - We need to understand what we should count

What do we count?

- Compilers don't warn Jeff Dean. Jeff Dean warns compilers.
- Jeff Dean builds his code before committing it, but only to check for compiler and linker bugs.
- Jeff Dean writes directly in binary. He then writes the source code as a documentation for other developers.
- Jeff Dean once shifted a bit so hard, it ended up on another computer.
- When Jeff Dean has an ergonomic evaluation, it is for the protection of his keyboard.
- gcc -O4 emails your code to Jeff Dean for a rewrite.
- When he heard that Jeff Dean's autobiography would be exclusive to the platform, Richard Stallman bought a Kindle.
- Jeff Dean puts his pants on one leg at a time, but if he had more legs, you'd realize the algorithm is actually only $O(\log n)$

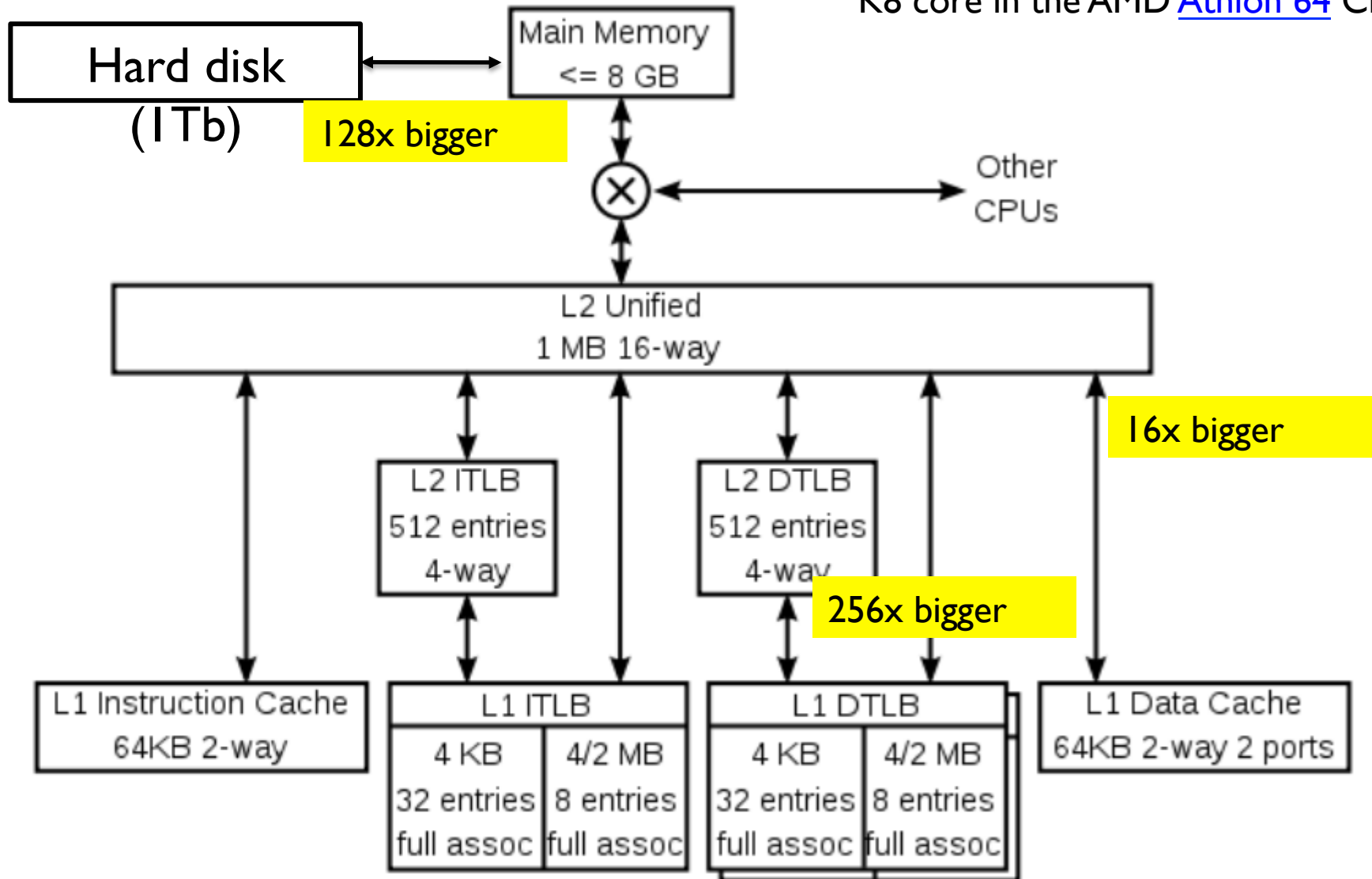


Numbers (Jeff Dean says) Everyone Should Know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

A typical CPU (not to scale)

K8 core in the AMD [Athlon 64](#) CPU

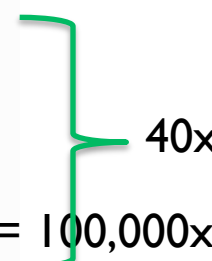


A typical disk



Numbers (Jeff Dean says) Everyone Should Know

L1 cache reference	0.5 ns	
Branch mispredict	5 ns	
L2 cache reference	7 ns	≈ 10x
Mutex lock/unlock	100 ns	
Main memory reference	100 ns	≈ 15x
Compress 1K bytes with Zippy	10,000 ns	
Send 2K bytes over 1 Gbps network	20,000 ns	
Read 1 MB sequentially from memory	250,000 ns	
Round trip within same datacenter	500,000 ns	
Disk seek	10,000,000 ns	
Read 1 MB sequentially from network	10,000,000 ns	≈ 100,000x
Read 1 MB sequentially from disk	30,000,000 ns	
Send packet CA→Netherlands→CA	150,000,000 ns	



40x

What do we count?

- Compilers don't warn Jeff Dean. Jeff Dean warns compilers.
-
- Memory access/instructions are *qualitatively different* from disk access
- Seeks are *qualitatively different* from sequential reads on disk
- Cache, disk fetches, etc work best when you stream through data *sequentially*
- Best case for data processing: stream through the data *once in sequential order*, as it's found on disk.



Other lessons -?



Encoding Your Data

- CPUs are fast, memory/bandwidth are precious, ergo...
 - Variable-length encodings
 - Compression
 - Compact in-memory representations
 - Compression very important^{*} aspect of many systems
 - inverted index posting list formats
 - storage systems for persistent data
- * but not important enough for this class's assignments....

What this course **is**

- Overview of the current “field” of data science and current frameworks
- First-hand experience with developing algorithms for large datasets
 - Hadoop, Spark
 - Deployment on Amazon EC2
- Emphasis on software engineering principles

What this course is **not**

- Introduction to programming
 - **Must** know Java
- Introduction to statistics and linear algebra
 - Self-evaluation on course website
- I will help with git and BitBucket
- I will help with Hadoop and Spark
- I will help with stats and linear algebra

Administrivia

- Office Hours: **[TBD]**
- Mailing list: csci6900-sl5@listserv.cc.uga.edu
- Course website: http://cobweb.cs.uga.edu/~squinn/mmd_sl5/
- Shannon Quinn (that's me)
 - 2008: Graduated from Georgia Tech [go Jackets!] in Computer Science (B.S.)
 - 2010: Graduated from Carnegie Mellon in Computational Biology (M.S.)
 - 2014: Graduated from University of Pittsburgh in Computational Biology (Ph.D.)
 - Worked at IBM, Google

Administrivia

- Programming Language:
 - Java and Hadoop
 - Scala / Python / Java and Spark
 - Most assignments will not use anything else
- Resources:
 - Your desktop/laptop
 - GSRC Hadoop virtual cluster
 - Getting this set up now...stay tuned
 - Amazon Elastic Cloud
 - Amazon EC2 [<http://aws.amazon.com/ec2/>]
 - Allocation: \$100 worth of time per student

Grading breakdown

- 40% assignments
 - Triweekly programming assignments
 - Not a lot of lines of code, but it will take you time to get them right
 - There are 4 possible assignments, you only need to do 3
- 35% project
 - 5-week project at end of course
 - I strongly encourage groups of 2
- 25% midterm
- 10% student research presentations

Coding

- All assignments will be committed to our team page on BitBucket
 - <https://bitbucket.org/csci6900-sl5/>
 - Concurrent versioning system: git
 - I want to see progress!
- First two assignments: Java and Hadoop
- Second two assignments: Spark
 - Spark has Python, Scala, and Java handles

Midterm

- Come to lecture
 - (that's not the midterm, but if you come to lecture, the midterm will be easy)

Student research presentations

- Each student presents once over the course of the semester

Basic idea:

1. Pick a paper from the “big data” literature
2. Prepare a 30-40 minute presentation
3. Lead a 20-30 minute discussion
4. ???
5. Profit!

Project

- More later
 - We will add a page with pointers to datasets and ideas for projects
 - Lots about scalable ML is still not well-understood so there's lots of opportunities for a meaningful study

To-do lists

YOU!

- Install git
- Create an account on BitBucket
- Email me your account name so I can add you to the BitBucket team
- Check out the “Administration” repository on BitBucket, and edit the STUDENT_LECTURES.md file to sign up for a presentation slot
- Check the “MAILING_LIST.md” file to ensure your information is correct

Me

- Post suggested papers for student presentations on website
- Post updated syllabus on website

Questions?

