Exam Review Session

William Cohen

General hints in studying

- Understand what you've done and *why*
  - There will be questions that test your *understanding* of the techniques implemented
    - why will/won't this shortcut work?
    - what does the analysis say about the method?
- We're mostly about learning meets computation here
  - there won't be many "pure 601" questions

General hints in studying

- Techniques covered in class but not assignments:
  - When/where/how to use them
  - That usually includes understanding the *analytic* results presented in class
  - Eg:
    - is the lazy regularization update an approximation or not? when does it help? when does it not help?

General hints in studying

- What about assignments you haven't done?
  - You should **read through the assignments** and be familiar with the algorithms being implemented
- There won't be questions about programming details that you could look up on line
  - but you should know how architectures like Hadoop work (eg, when and where they communicate)
  - you should be able to sketch out simple map-reduce algorithms
  - No spark, but you should be able to read workflow operators and discuss how they'd be implemented
    - how would you do a groupByKey in Hadoop?

General hints in studying

- There are not ~~detailed~~ questions on the guest speakers or the student projects (this year)
- If you use a previous year's exam as guidance
  - the topics are a little different each year

General hints in exam taking

- You can bring in one 8 ½ by 11” sheet (front and back)
- Look over everything quickly and skip around
  - probably nobody will know everything on the test
- If you’re not sure what we’ve got in mind: state your *assumptions* clearly in your answer.
  - There’s room for this even on true/false
- If you look at a question and don’t know the answer:
  - we probably haven’t told you the answer
  - but we’ve told you enough to work it out

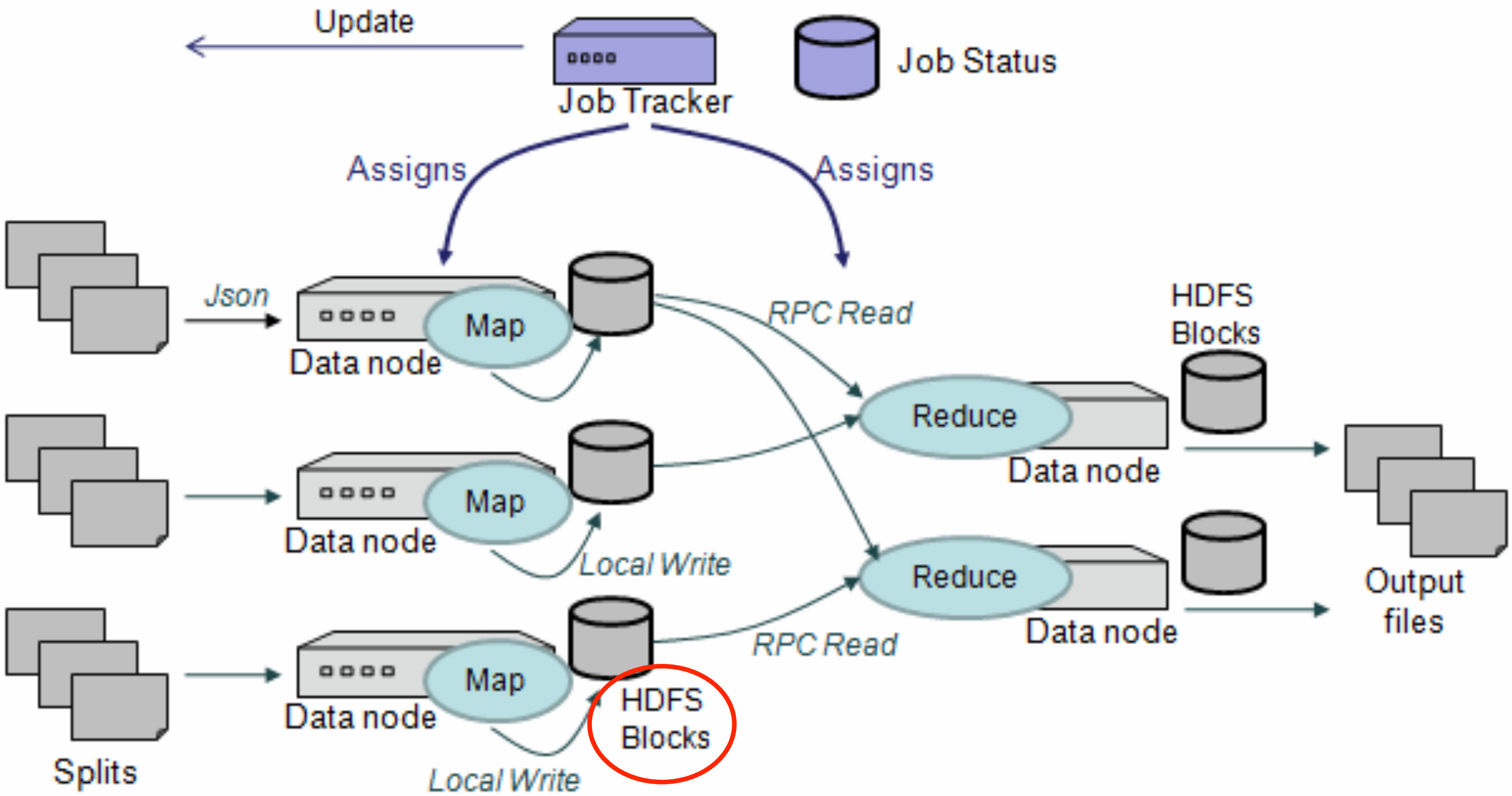
  - imagine arguing for some answer and see if you like it

Outline – major topics

- Hadoop
  - stream-and-sort is how I ease you into that, not really a topic on its own
- Parallelizing learners (perceptron, LDA, …)
- Hash kernels and streaming SGD
- Distributed SGD for Matrix Factorization
- Randomized Algorithms
- Graph Algorithms: PR, PPR, SSL on graphs
  - no assignment != not on test

- Some of these are easier to ask questions about than others.

HADOOP



What data gets lost if the job tracker is rebooted?
If I have a 1Tb file and shard it 1000 ways will it take longer than sharding it 10 ways?
Where should a combiner run?

```
$ hadoop fs -ls rcv1/small/sharded
Found 10 items
-rw-r--r--   3 …  606405 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00000
-rw-r--r--   3 … 1347611 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00001
-rw-r--r--   3 …  939307 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00002
-rw-r--r--   3 … 1284062 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00003
-rw-r--r--   3 … 1009890 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00004
-rw-r--r--   3 … 1206196 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00005
-rw-r--r--   3 … 1384658 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00006
-rw-r--r--   3 … 1299698 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00007
-rw-r--r--   3 …  928752 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00008
-rw-r--r--   3 …  806030 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00009

$ hadoop fs -tail rcv1/small/sharded/part-00005
weak as the arrival of arbitraged cargoes from the West has put the local market under pressure…
M14,M143,MCAT    The Brent crude market on the Singapore International …
```

Where is this data? How many disks is it on?
If I set up a directory on /afs that looks the same will it work the same? what about a local disk?

Hadoop job_201301231150_0778 on hadoopjt

User: wcohen
Job Name: streamjob6055532903853567038.jar
Job File: hdfs://hdfsname.opencloud/l/a2/scratch/hadoop-data/global/mapred/system/job_201301231150_0778/job.xml
Job Setup: Successful
Status: Failed
Started at: Wed Jan 30 11:46:47 EST 2013
Failed at: Wed Jan 30 11:47:28 EST 2013
Failed in: 41sec
Job Cleanup: Successful
Black-listed TaskTrackers: 2
Job Scheduling information: 5 running map tasks using 5 map slots, 0 running reduce tasks using 0 reduce slots.

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	10	0	0	0	10	35 / 5
reduce	[illegible]00%	10	0	0	0	10	0 / 0

	Counter	Map	Reduce	Total
Job Counters	Rack-local map tasks	0	0	38
	Launched map tasks	0	0	40
	Data-local map tasks	0	0	2
	Failed map tasks	0	0	1

Map Completion Graph - close

Hadoop map task list for job_201301231150_0778 on hadoop

All Tasks

Task	Complete	Status	Start Time	Finish Time	Errors
task_201301231150_0778_m_000000	0.00%		30-Jan-2013 11:47:01	30-Jan-2013 11:47:25 (24sec)	java.lang.RuntimeException: PipeMa at org.apache.hadoop.strea at org.apache.hadoop.strea at org.apache.hadoop.strea at org.apache.hadoop.mapre at org.apache.hadoop.strea at org.apache.hadoop.mapre at org.apache.hadoop.mapre at org.apache.hadoop.mapre java.lang.RuntimeException: PipeMa at org.apache.hadoop.strea at org.apache.hadoop.strea at org.apache.hadoop.strea at org.apache.hadoop.mapre at org.apache.hadoop.strea at org.apache.hadoop.mapre at org.apache.hadoop.mapre at org.apache.hadoop.mapre java.lang.RuntimeException: PipeMa at org.apache.hadoop.strea at org.apache.hadoop.strea at org.apache.hadoop.strea at org.apache.hadoop.mapre at org.apache.hadoop.strea at org.apache.hadoop.mapre at org.apache.hadoop.mapre at org.apache.hadoop.mapre java.lang.RuntimeException: PipeMa at org.apache.hadoop.strea

Job job_201301231150_0778

All Task Attempts

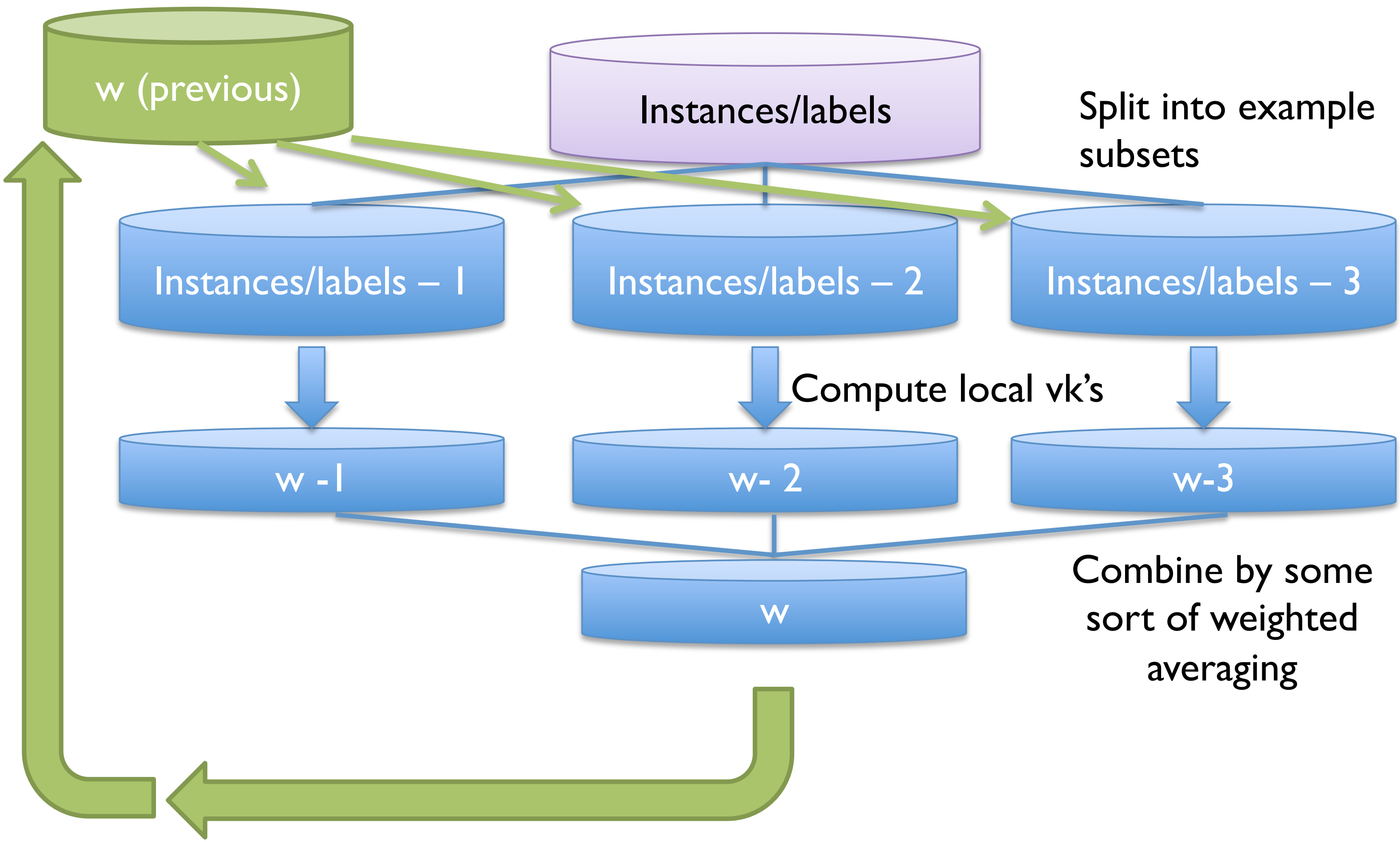
Task Attempts	Machine	Status	Progress	Start Time	Finish Time	Error
attempt_201301231150_0778_m_000000_0	/default-rack/cloud3u12.opencloud	FAILED	0.00%	30-Jan-2013 11:47:01	30-Jan-2013 11:47:06 (4sec)	java.
attempt_201301231150_0778_m_000000_1	/default-rack/cloud2u28.opencloud	FAILED	0.00%	30-Jan-2013 11:47:07	30-Jan-2013 11:47:11 (4sec)	java.

me	Errors	Task Logs	Counters	Actions
2013	java.lang.RuntimeException: PipeMapRed.waitOutputThreads(): subprocess failed w[illegible]de 1 at org.apache.hadoop.streaming.PipeMapRed.waitOutputThreads(PipeMapRed.j[illegible]1) at org.apache.hadoop.streaming.PipeMapRed.mapRedFinished(PipeMapRed.java:[illegible] at org.apache.hadoop.streaming.PipeMapper.close(PipeMapper.java:132) at org.apache.hadoop.mapred.MapRunner.run(MapRunner.java:57) at org.apache.hadoop.streaming.PipeMapRunner.run(PipeMapRunner.java:36) at org.apache.hadoop.mapred.MapTask.runOldMapper(MapTask.java:358) at org.apache.hadoop.mapred.MapTask.run(MapTask.java:307) at org.apache.hadoop.mapred.Child.main(Child.java:170)	Last 4KB Last 8KB All	1	
2013	java.lang.RuntimeException: PipeMapRed.waitOutputThreads(): subprocess failed with code 1 at org.apache.hadoop.streaming.PipeMapRed.waitOutputThreads(PipeMapRed.java:311) at org.apache.hadoop.streaming.PipeMapRed.mapRedFinished(PipeMapRed.java:540) at org.apache.hadoop.streaming.PipeMapper.close(PipeMapper.java:132) at org.apache.hadoop.mapred.MapRunner.run(MapRunner.java:57) at org.apache.hadoop.streaming.PipeMapRunner.run(PipeMapRunner.java:36) at org.apache.hadoop.mapred.MapTask.runOldMapper(MapTask.java:358) at org.apache.hadoop.mapred.MapTask.run(MapTask.java:307) at org.apache.hadoop.mapred.Child.main(Child.java:170)	Last 4KB Last 8KB All	1	
2013	java.lang.RuntimeException: PipeMapRed.waitOutputThreads(): subprocess failed with code 1 at org.apache.hadoop.streaming.PipeMapRed.waitOutputThreads(PipeMapRed.java:311) at org.apache.hadoop.streaming.PipeMapRed.mapRedFinished(PipeMapRed.java:540) at org.apache.hadoop.streaming.PipeMapper.close(PipeMapper.java:132) at org.apache.hadoop.mapred.MapRunner.run(MapRunner.java:57) at org.apache.hadoop.streaming.PipeMapRunner.run(PipeMapRunner.java:36) at org.apache.hadoop.mapred.MapTask.runOldMapper(MapTask.java:358) at org.apache.hadoop.mapred.MapTask.run(MapTask.java:307) at org.apache.hadoop.mapred.Child.main(Child.java:170)	Last 4KB Last 8KB All	1	

Why do I see this same error over and over again?

PARALLEL LEARNERS

Parallelizing perceptrons – take 2



A theorem

Theorem 3. *Assume a training set $\mathcal{T}$ is separable by margin γ. Let $k_{i,n}$ be the number of mistakes that occurred on shard i during the nth epoch of training. For any N, when training the perceptron with iterative parameter mixing (Figure 3),*

$$\sum_{n=1}^{N}\sum_{i=1}^{S}\mu_{i,n}k_{i,n} \leq \frac{R^2}{\gamma^2}$$

I probably won't ask about the proof, but I could definitely ask about the theorem.

Corollary: if we weight the vectors uniformly, then the number of mistakes is still bounded.

I.e., this is "enough communication" to guarantee convergence.

Distributed Training Strategies for the Structured Perceptron

Ryan McDonald **Keith Hall** **Gideon Mann**
Google, Inc., New York / Zurich
{ryanmcd|kbhall|gmann}@google.com

NAACL 2010

What does the word "structured" mean here? why is it important? would the results be better or worse with a regular perceptron?

STREAMING SGD

Learning as optimization for regularized logistic regression

what if you had a different update for the regularizer?

- Algorithm: $$w^j = w^j + \lambda(y-p)x^j - \lambda 2\mu w^j$$

Time goes from O(nT) to O(mVT) where
- n = number of non-zero entries,
- m = number of examples
- V = number of features
- T = number of passes over data

1. Initialize a hashtable W
2. For $t = 1, \ldots, T$
   - For each example $\mathbf{x}_i, y_i$:
     - Compute the prediction for $\mathbf{x}_i$:
       $$p_i = \frac{1}{1 + \exp(-\sum_{j:x_i^j > 0} x_i^j w^j)}$$
     - For each ~~non-zero~~ feature of $\mathbf{x_i}$ with index j and value x^j:
       * If j is not in W, set $W[j] = 0$.
       * Set $W[j] = W[j] \boxed{+ \lambda(y-p)x^j - \lambda 2\mu w^j}$
3. Output the hash table W.

Formalization of the "Hash Trick":

First: Review of Kernels

What is it? how does it work? what aspects of performance does it help? What did we say about it formally?

RANDOMIZED ALGORITHMS

Randomized Algorithms

- Hash kernels
- Countmin sketch
- Bloom filters
- LSH

- **What** are they, and what are they used for? **When** would you use which one?
- **Why** do they work - ie, what analytic results have we looked at?

Bloom filters - review

- An implementation
  - Allocate M bits, bit[0]…,bit[1-M]
  - Pick K hash functions hash(1,2),hash(2,s),….
    - E.g: hash(i,s) = hash(s+ randomString[i])
  - To add string s:
    - For i=1 to k, set bit[hash(i,s)] = 1
  - To check contains(s):
    - For i=1 to k, test bit[hash(i,s)]
    - Return "true" if they're all set; otherwise, return "false"
  - We'll discuss how to set M and K soon, but for now:
    - Let M = 1.5*maxSize *// less than two bits per item!*
    - Let K = 2*log(1/p) *// about right with this M*

Bloom filters

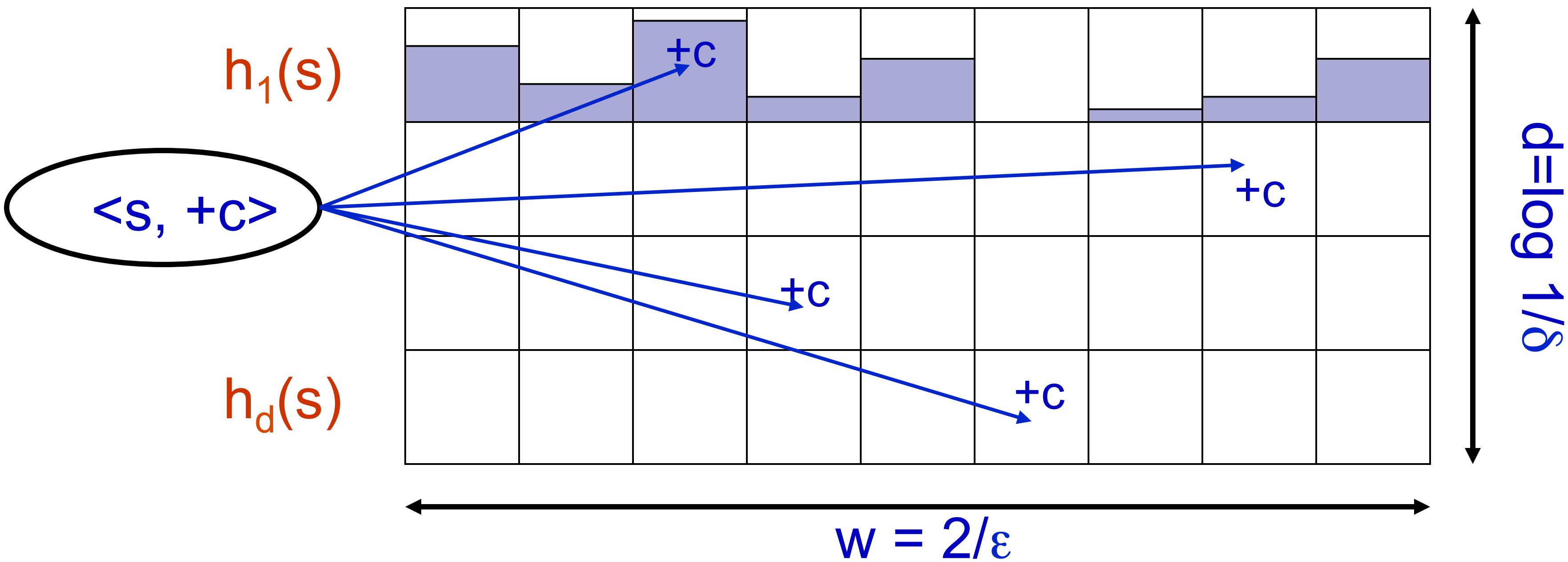
- An example application
  - discarding rare features from a classifier
  - seldom hurts much, can speed up experiments
- Scan through data once and check each *w*:
  - if bf1.contains(*w*):
    - if bf2.contains(w): bf3.add(*w*)
    - else bf2.add(w)
  - else bf1.add(*w*)
- Now:
  - bf2.contains(w) ⇔ w appears >= 2x
  - bf3.contains(w) ⇔ w appears >= 3x
- Then train, ignoring words not in bf3

which needs more storage, bf1 or bf3?
(same false positive rate)

Bloom filters

- Here’s two ideas for using Bloom filters for learning from sparse binary examples:
  - compress every example with a BF
  - either
    - use each bit of the BF as a feature for a classifier
    - or: reconstruct the example at training time and train an ordinary classifier
  - pros and cons?

CM Sketch Structure



- Each string is mapped to one bucket per row
- Estimate A[j] by taking $\min_k \{ CM[k,h_k(j)] \}$
- Errors are always over-estimates
- Sizes: $d=\log 1/\delta$, $w=2/\varepsilon$ ➔ error is usually less than $\varepsilon||A||_1$

CM Sketch Guarantees

- *[Cormode, Muthukrishnan'04]* CM sketch guarantees approximation error on point queries less than $\varepsilon||A||_1$ in space $O(1/\varepsilon \log 1/\delta)$
  - Probability of more error is less than $1-\delta$

- This is sometimes enough:
  - Estimating a multinomial: if A[s] = Pr(s|…) then $||A||_1 = 1$
  - Multiclass classification: if $A_x[s]$ = Pr(x in class s) then $||A_x||_1$ is probably small, since most x's will be in only a few classes

An Application of a Count-Min Sketch

- Problem: find the semantic orientation of a work (positive or negative) using a large corpus.
- Idea:
  - positive words co-occur more frequently than expected near positive words; likewise for negative words
  - so pick a few pos/neg seeds and compute

x appears near y

$$\mathrm{pmi}(x;y) \equiv \log \frac{\boxed{p(x,y)}}{p(x)p(y)}$$

$$\mathrm{SO(w)} = \sum_{\mathrm{p}\in\mathrm{Pos}} PMI(\mathrm{p},\mathrm{w}) - \sum_{\mathrm{n}\in\mathrm{Neg}} PMI(\mathrm{n},\mathrm{w})$$

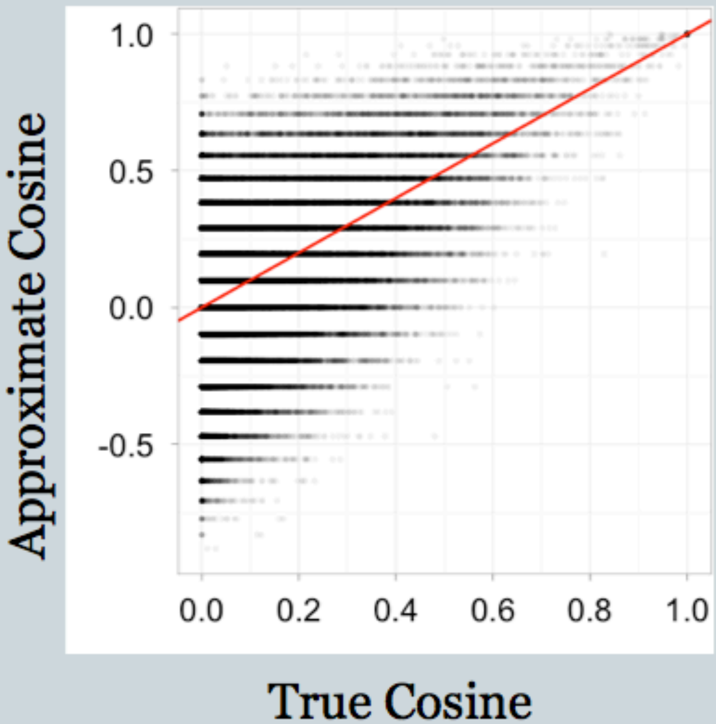
LSH: key ideas

- Goal:
  - map feature vector **x** to bit vector **bx**
  - ensure that **bx** preserves “similarity”
- Basic idea: use random projections of **x**
  - Repeat many times:
    - Pick a random hyperplane **r** by picking random weights for each feature (say from a Gaussian)
    - Compute the inner product of **r** with **x**
    - Record if **x** is “close to” **r** (**r.x**>=0)
      - the next bit in **bx**
    - Theory says that if **x’** and **x** have small cosine distance then **bx** and **bx’** will have small Hamming distance

LSH applications

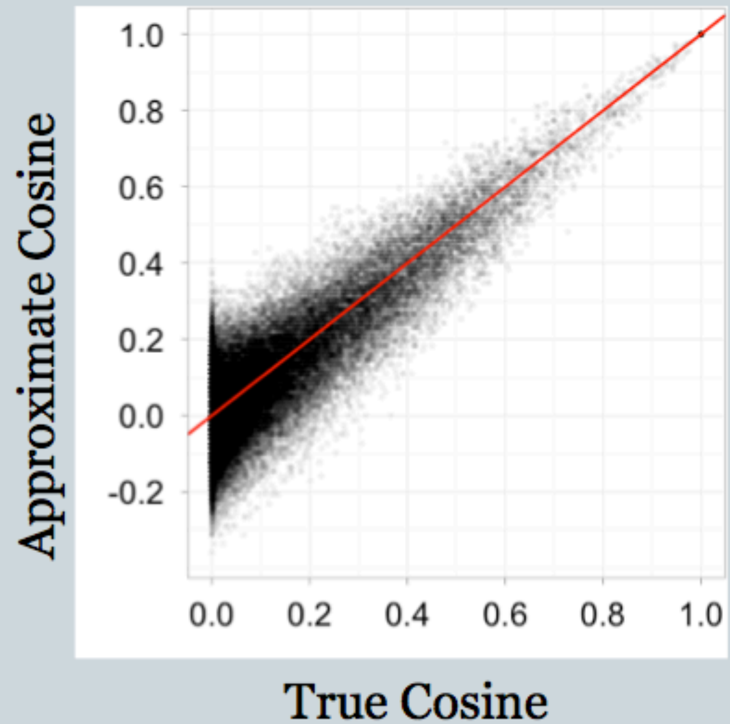
- Compact storage of data
  - and we can still compute similarities
- LSH also gives very fast approximations:
  - approx nearest neighbor method
    - just look at other items with **bx'=bx**
    - also very fast nearest-neighbor methods for Hamming distance
  - very fast clustering
    - cluster = all things with same **bx** vector

32 bit signatures



Cheap

256 bit signatures



Accurate

LSH

- What are some other ways of using LSH?
- What are some other ways of using CountMin sketchs?

GRAPH ALGORITHMS

Graph Algorithms

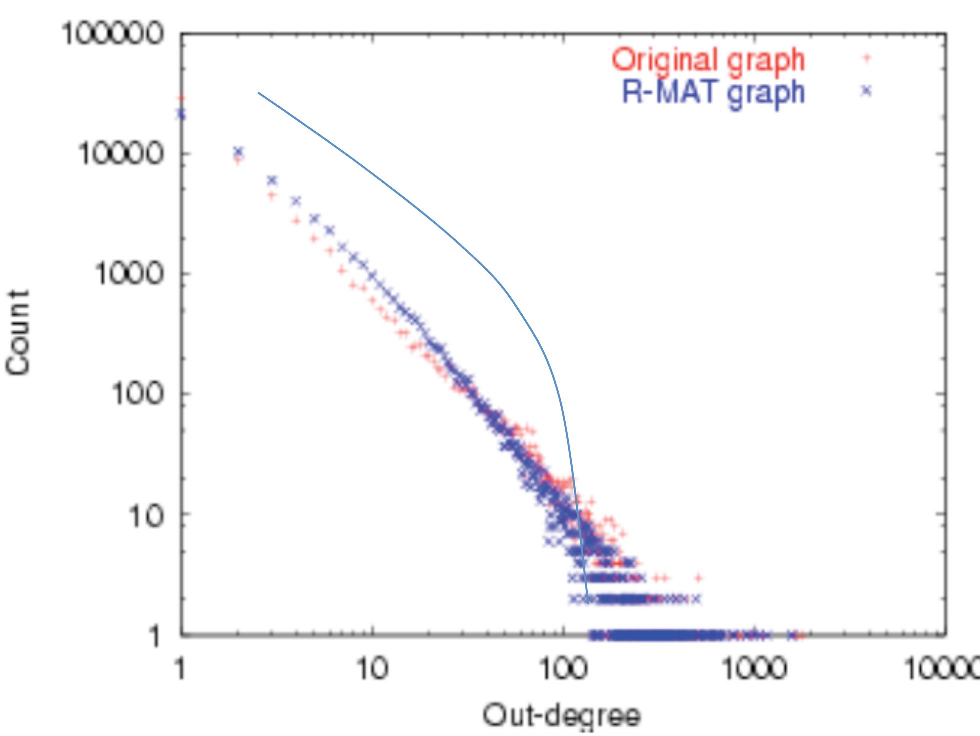
- The "standard" way of computing PageRank, iteratively, using the power method
- Properties of large data graphs
  - ...
- The "subsampling problem"
- APR algorithm - how and *why* it works
  - when we'd use it
  - what the analysis says
- SSL on graphs
  - example algorithms, example use of CM Sketches

	network	type	n	m	z	ℓ	α	$C^{(1)}$	$C^{(2)}$
social	film actors	undirected	449 913	25 516 482	113.43	3.48	2.3	0.20	0.78
	company directors	undirected	7 673	55 392	14.44	4.60	–	0.59	0.88
	math coauthorship	undirected	253 339	496 489	3.92	7.57	–	0.15	0.34
	physics coauthorship	undirected	52 909	245 300	9.27	6.19	–	0.45	0.56
	biology coauthorship	undirected	1 520 251	11 803 064	15.53	4.92	–	0.088	0.60
	telephone call graph	undirected	47 000 000	80 000 000	3.16		2.1		
	email messages	directed	59 912	86 300	1.44	4.95	1.5/2.0		0.16
	email address books	directed	16 881	57 029	3.38	5.22	–	0.17	0.13
	student relationships	undirected	573	477	1.66	16.01	–	0.005	0.001
	sexual contacts	undirected	2 810				3.2		
information	WWW `nd.edu`	directed	269 504	1 497 135	5.55	11.27	2.1/2.4	0.11	0.29
	WWW Altavista	directed	203 549 046	2 130 000 000	10.46	16.18	2.1/2.7		
	citation network	directed	783 339	6 716 198	8.57		3.0/–		
	Roget's Thesaurus	directed	1 022	5 103	4.99	4.87	–	0.13	0.15
	word co-occurrence	undirected	460 902	17 000 000	70.13		2.7		0.44
technological	Internet	undirected	10 697	31 992	5.98	3.31	2.5	0.035	0.39
	power grid	undirected	4 941	6 594	2.67	18.99	–	0.10	0.080
	train routes	undirected	587	19 603	66.79	2.16	–		0.69
	software packages	directed	1 439	1 723	1.20	2.42	1.6/1.4	0.070	0.082
	software classes	directed	1 377	2 213	1.61	1.51	–	0.033	0.012
	electronic circuits	undirected	24 097	53 248	4.34	11.05	3.0	0.010	0.030
	peer-to-peer network	undirected	880	1 296	1.47	4.28	2.1	0.012	0.011
biological	metabolic network	undirected	765	3 686	9.64	2.56	2.2	0.090	0.67
	protein interactions	undirected	2 115	2 240	2.12	6.80	2.4	0.072	0.071
	marine food web	directed	135	598	4.43	2.05	–	0.16	0.23
	freshwater food web	directed	92	997	10.84	1.90	–	0.20	0.087
	neural network	directed	307	2 359	7.68	3.97	–	0.18	0.28

Degree distribution

- Plot cumulative degree
  - X axis is degree
  - Y axis is #nodes that have degree at least *k*
- Typically use a log-log scale
  - Straight lines are a power law; normal curve dives to zero at some point
  - Left: trust network in epinions web site from Richardson & Domingos

count



Homophily

- Another def'n: excess edges between common neighbors of *v*

$$CC(v) = \frac{\#\,\text{triangles connected to } v}{\#\,\text{pairs connected to } v}$$

$$CC(V,E) = \frac{1}{|V|}\sum_{v} CC(v)$$

$$\boxed{CC'(V,E) = \frac{\#\,\text{triangles in graph}}{\#\,\text{length 3 paths in graph}}}$$

Approximate PageRank: Algorithm

`ApproximatePageRank` (v, α, ϵ):

1. Let $p = \vec{0}$, and $r = \chi_v$.
2. While $\max_{u \in V} \frac{r(u)}{d(u)} \geq \epsilon$:
   (a) Choose any vertex u where $\frac{r(u)}{d(u)} \geq \epsilon$.
   (b) Apply $\texttt{push}_u$ at vertex u, updating p and r.
3. Return p, which satisfies $p = \mathrm{apr}(\alpha, \chi_v, r)$ with $\max_{u \in V} \frac{r(u)}{d(u)} < \epsilon$.

$\texttt{push}_u(p, r)$:

1. Let $p' = p$ and $r' = r$, except for the following changes:
   (a) $p'(u) = p(u) + \alpha r(u)$.
   (b) $r'(u) = (1 - \alpha) r(u)/2$.
   (c) For each v such that $(u, v) \in E$: $\quad r'(v) = r(v) + (1 - \alpha) r(u)/(2d(u))$.
2. Return (p', r').

Approximate PageRank: Key Idea

By definition PageRank is fixed point of:

$$\mathrm{pr}(\alpha, s) = \alpha s + (1-\alpha)\mathrm{pr}(\alpha, s)W,$$

Claim:

$$\mathrm{pr}(\alpha, s) = \alpha s + (1-\alpha)\mathrm{pr}(\alpha, sW).$$

Recursively compute PageRank of "neighbors of *s*" *(=sW)*, then adjust

Key idea in apr:

- do this "recursive step" repeatedly
- focus on nodes where finding PageRank from neighbors will be useful

Analysis

Lemma 1. *Let p' and r' be the result of the operation* $\mathbf{push}_u$ *on p and r. Then*

$$p' + \mathrm{pr}(\alpha, r') = p + \mathrm{pr}(\alpha, r).$$

Proof of Lemma 1. After the push operation, we have

$$\begin{aligned} p' &= p + \alpha r(u)\chi_u. \\ r' &= r - r(u)\chi_u + (1-\alpha)r(u)\chi_u W. \end{aligned}$$

Using equation (5),

linearity

$$\begin{aligned} p + \mathrm{pr}(\alpha, r) &= p + \mathrm{pr}(\alpha, r - r(u)\chi_u) + \underline{\mathrm{pr}(\alpha, r(u)\chi_u)} \\ &= p + \mathrm{pr}(\alpha, r - r(u)\chi_u) + \underline{[\alpha r(u)\chi_u + (1-\alpha)\mathrm{pr}(\alpha, r(u)\chi_u W)]} \\ &= [p + \alpha r(u)\chi_u] + \mathrm{pr}(\alpha, [r - r(u)\chi_u + (1-\alpha)r(u)\chi_u W]) \\ &= p' + \mathrm{pr}(\alpha, r'). \end{aligned}$$

re-group & linearity

$pr(\alpha, r - r(u)\chi_u) + (1\text{-}\alpha)\, pr(\alpha, r(u)\chi_u W) = pr(\alpha, r - r(u)\chi_u + (1\text{-}\alpha)\ r(u)\chi_u W)$

$$\mathrm{pr}(\alpha, s) = \alpha s + (1-\alpha)\mathrm{pr}(\alpha, sW). \qquad (5)$$

Q/A