# LDA FOR BIG DATA

# LDA for Big Data - Outline

- Quick review of LDA model
  - clustering words-in-context
- Parallel LDA ~= IPM
- Fast sampling tricks for LDA
  - Sparsified sampler
  - Alias table
  - Fenwick trees
- LDA for text → LDA-like models for graphs

# Recap: The LDA Topic Model

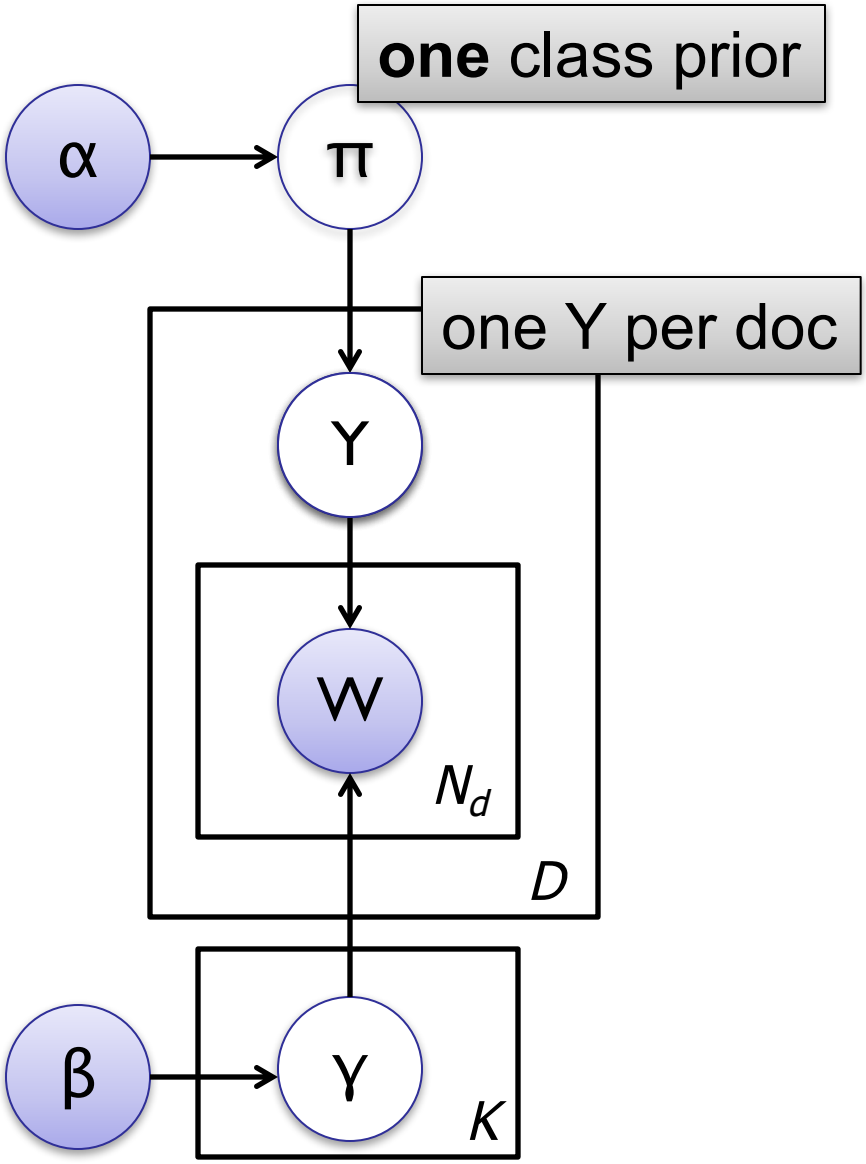
**Latent Dirichlet Allocation**

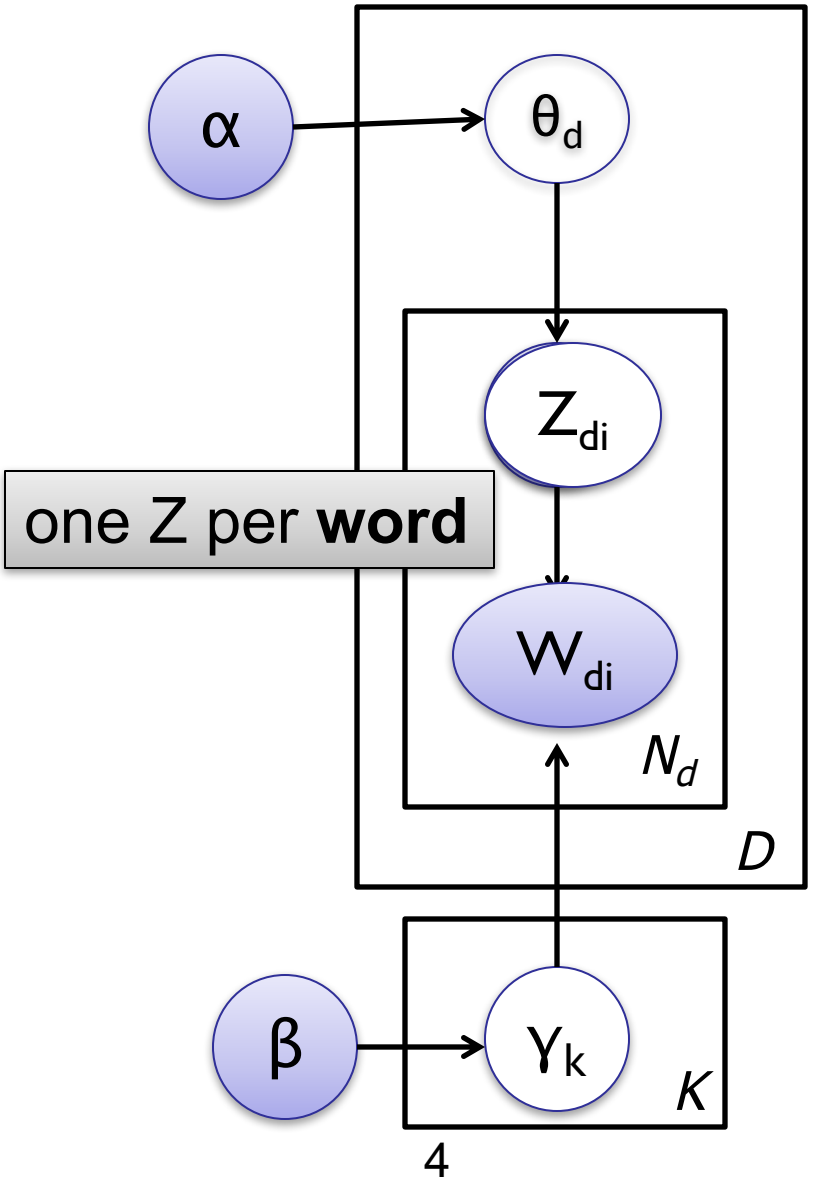
**David M. Blei**
*Computer Science Division*
*University of California*
*Berkeley, CA 94720, USA*

**Andrew Y. Ng**
*Computer Science Department*
*Stanford University*
*Stanford, CA 94305, USA*

**Michael I. Jordan**
*Computer Science Division and Department of Statistics*
*University of California*
*Berkeley, CA 94720, USA*

# Unsupervised NB vs LDA



- LDA topics: top words w by Pr(w|Z=k)

| Z=13 | Z=22 | Z=27 | Z=19 |
|---|---|---|---|
| "Arts" | "Budgets" | "Children" | "Education" |
| NEW | MILLION | CHILDREN | SCHOOL |
| FILM | TAX | WOMEN | STUDENTS |
| SHOW | PROGRAM | PEOPLE | SCHOOLS |
| MUSIC | BUDGET | CHILD | EDUCATION |
| MOVIE | BILLION | YEARS | TEACHERS |
| PLAY | FEDERAL | FAMILIES | HIGH |
| MUSICAL | YEAR | WORK | PUBLIC |
| BEST | SPENDING | PARENTS | TEACHER |
| ACTOR | NEW | SAYS | BENNETT |
| FIRST | STATE | FAMILY | MANIGAT |
| YORK | PLAN | WELFARE | NAMPHY |
| OPERA | MONEY | MEN | STATE |
| THEATER | PROGRAMS | PERCENT | PRESIDENT |
| ACTRESS | GOVERNMENT | CARE | ELEMENTARY |
| LOVE | CONGRESS | LIFE | HAITI |

- LDA's view of a document

Mixed membership model

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

"Arts" "Budgets" "Children" "Education"

# LDA and (Collapsed) Gibbs Sampling

- Gibbs sampling – works for *any* directed model!
  - Applicable when joint distribution is hard to evaluate but conditional distribution is known
  - Sequence of samples comprises a Markov Chain
  - Stationary distribution of the chain is the joint distribution

1. Initialise $x_{0,1:n}$.
2. For $i = 0$ to $N - 1$
   - Sample $x_1^{(i+1)} \sim p(x_1 | x_2^{(i)}, x_3^{(i)}, \ldots, x_n^{(i)})$.
   - Sample $x_2^{(i+1)} \sim p(x_2 | x_1^{(i+1)}, x_3^{(i)}, \ldots, x_n^{(i)})$.

   $\vdots$

   - Sample $x_j^{(i+1)} \sim p(x_j | x_1^{(i+1)}, \ldots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, \ldots, x_n^{(i)})$.

   $\vdots$

   - Sample $x_n^{(i+1)} \sim p(x_n | x_1^{(i+1)}, x_2^{(i+1)}, \ldots x_{n-1}^{(i+1)})$.

Key capability: estimate distribution of **one** latent variables given **the other latent variables** and observed variables.

# Recap: Collapsed Sampling for LDA

Pr(Z|E+)

Pr(E−|Z)

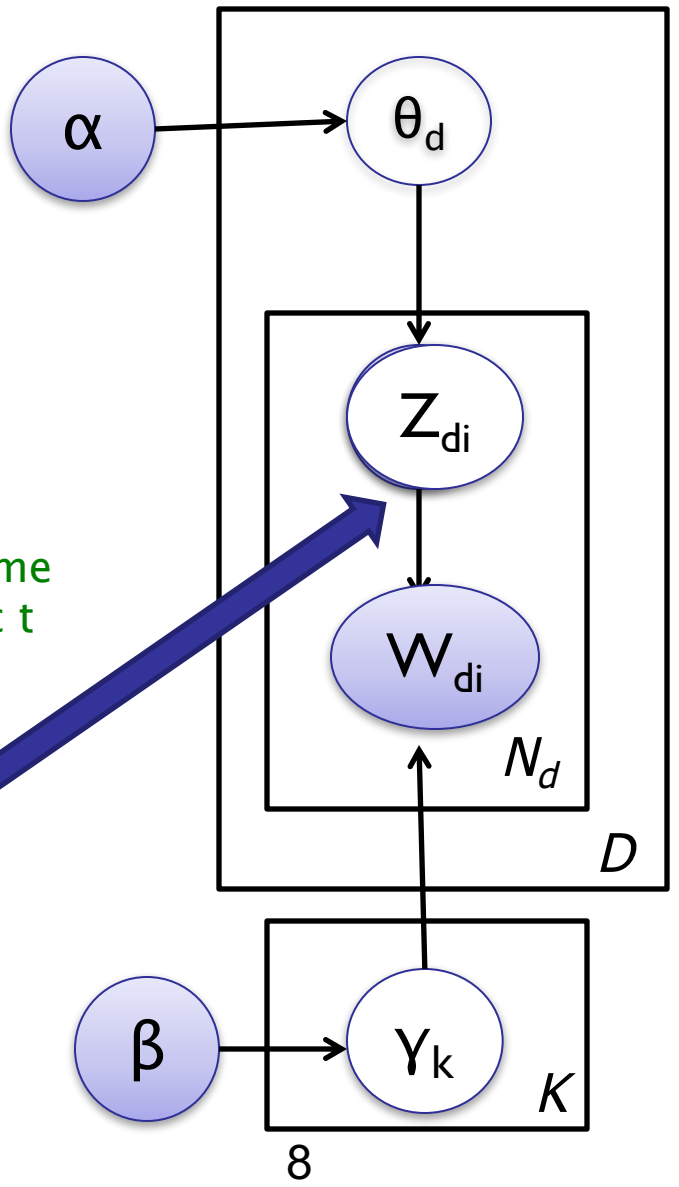
$$P(z = t|w) \quad \propto \quad (\alpha_t + n_{t|d}) \frac{\beta + n_{w|t}}{\beta V + n_{\cdot|t}}.$$

"fraction" of time
*Z=t* in doc *d*

fraction of time
W=w in topic t

ignores a detail – counts should not include the $Z_{di}$ being sampled

Only sample the Z's



# PARALLEL LDA

# Distributed Algorithms for Topic Models

**David Newman** NEWMAN@UCI.EDU
**Arthur Asuncion** ASUNCION@ICS.UCI.EDU
**Padhraic Smyth** SMYTH@ICS.UCI.EDU
**Max Welling** WELLING@ICS.UCI.EDU
*Department of Computer Science*
*University of California, Irvine*
*Irvine, CA 92697, USA*

JMLR 2009

# Observation

- How much does the choice of *z* depend on the other *z's* in the same document?
  - quite a lot
- How much does the choice of *z* depend on the other *z's* in elsewhere in the corpus?
  - maybe not so much
  - depends on Pr(w|t) but that changes slowly
- Can we parallelize Gibbs and still get good results?

# Question

- Can we parallelize Gibbs sampling?
  - formally, no: every choice of $z$ depends on all the other $z$'s
  - Gibbs needs to be sequential
    - just like SGD

# What if you try and parallelize?

Split document/term matrix randomly and distribute to $p$ processors .. then run "Approximate Distributed LDA"

**Algorithm 1** AD-LDA

**repeat**
  **for** each processor $p$ in parallel **do**
    Copy global counts: $N_{wkp} \leftarrow N_{wk}$
    Sample $\mathbf{z}_p$ locally: LDA-Gibbs-Iteration($\mathbf{x}_p, \mathbf{z}_p, N_{kjp}, N_{wkp}, \alpha, \beta$)
  **end for**
  Synchronize
  Update global counts: $N_{wk} \leftarrow N_{wk} + \sum_p (N_{wkp} - N_{wk})$
**until** termination criterion satisfied

This is iterative parameter mixing

# What if you try and parallelize?

| | LDA | AD-LDA |
|---|---|---|
| Space | $N + K(D + W)$ | $\frac{1}{P}(N + KD) + KW$ |
| Time | $NK$ | $\frac{1}{P}NK + KW + C$ |

All-Reduce cost

Table 3: Space and time complexity of LDA and AD-LDA.

D=#docs W=#word(types) K=#topics N=words in corpus

| | KOS | NIPS | WIKIPEDIA | PUBMED | NEWSGROUPS |
|---|---|---|---|---|---|
| $D_{\text{train}}$ | 3,000 | 1,500 | 2,051,929 | 8,200,000 | 19500 |
| $W$ | 6,906 | 12,419 | 120,927 | 141,043 | 27,059 |
| $N$ | 467,714 | 2,166,058 | 344,941,756 | 737,869,083 | 2,057,207 |
| $D_{\text{test}}$ | 430 | 184 | - | - | 498 |

Table 2: Characteristics of data sets used in experiments.

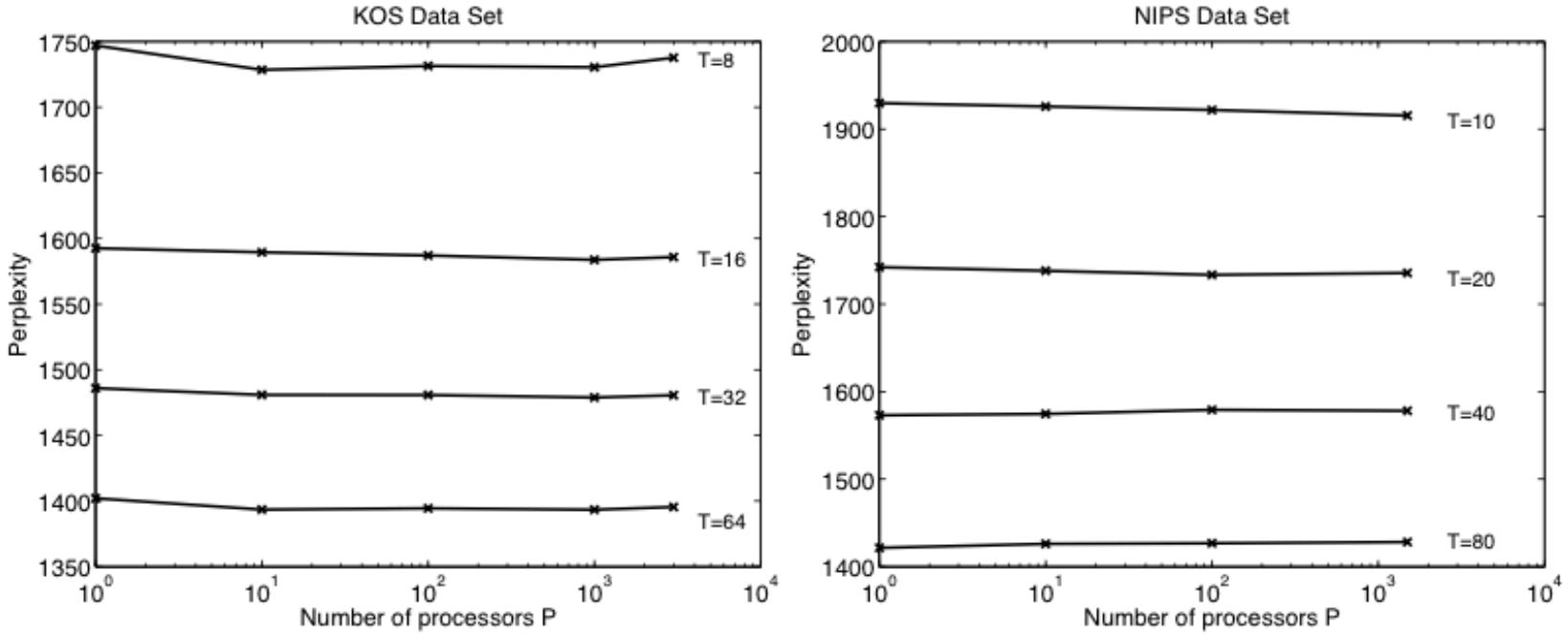
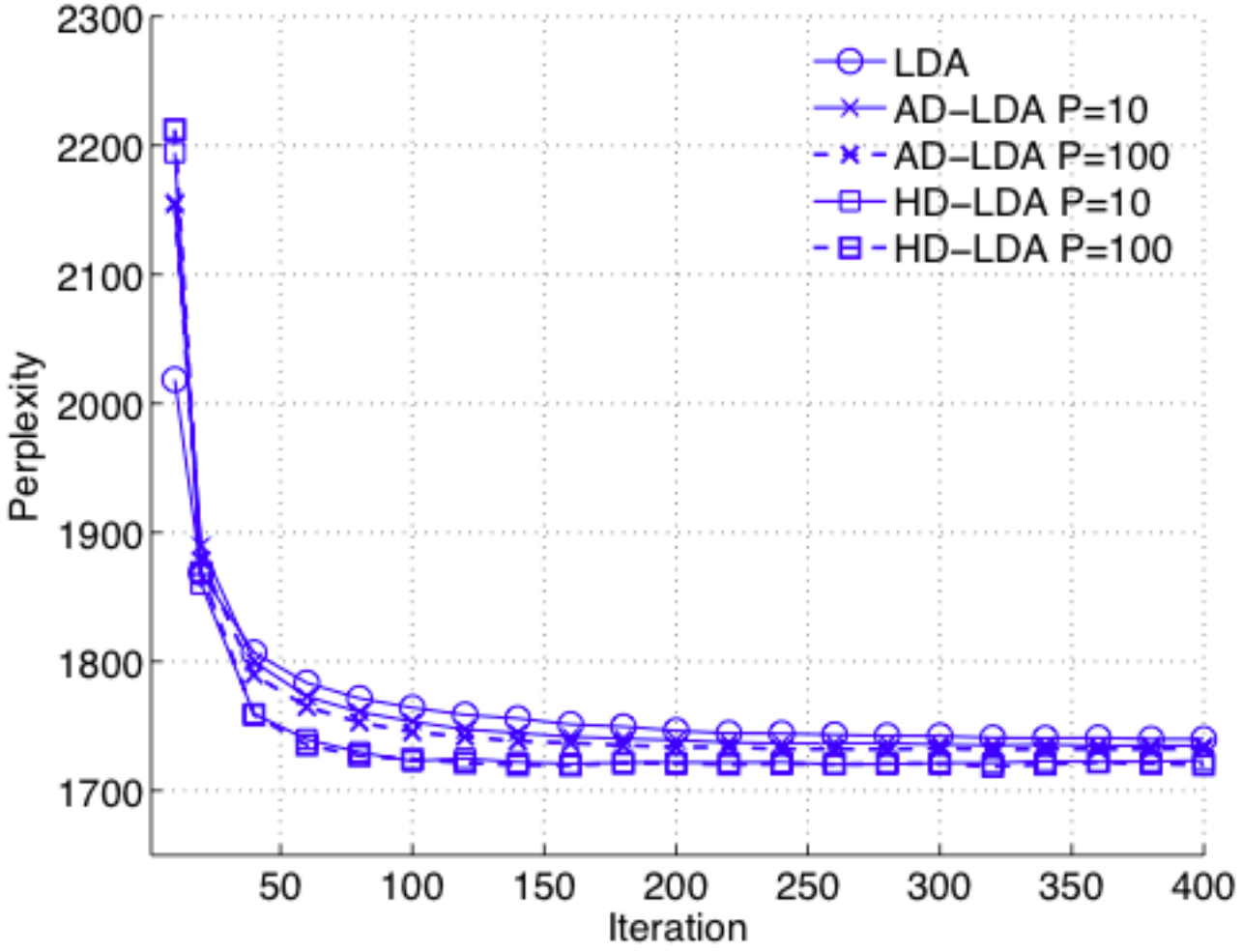



Figure 6: AD-LDA test perplexity versus number of processors up to the limiting case of number of processors equal to number of documents in collection. Left plot shows perplexity for KOS and right plot shows perplexity for NIPS.


2300
2200
2100
2000
1900
1800
1700
Perplexity
50
100
150
200
250
300
350
400
Iteration
LDA
AD–LDA P=10
AD–LDA P=100
HD–LDA P=10
HD–LDA P=100

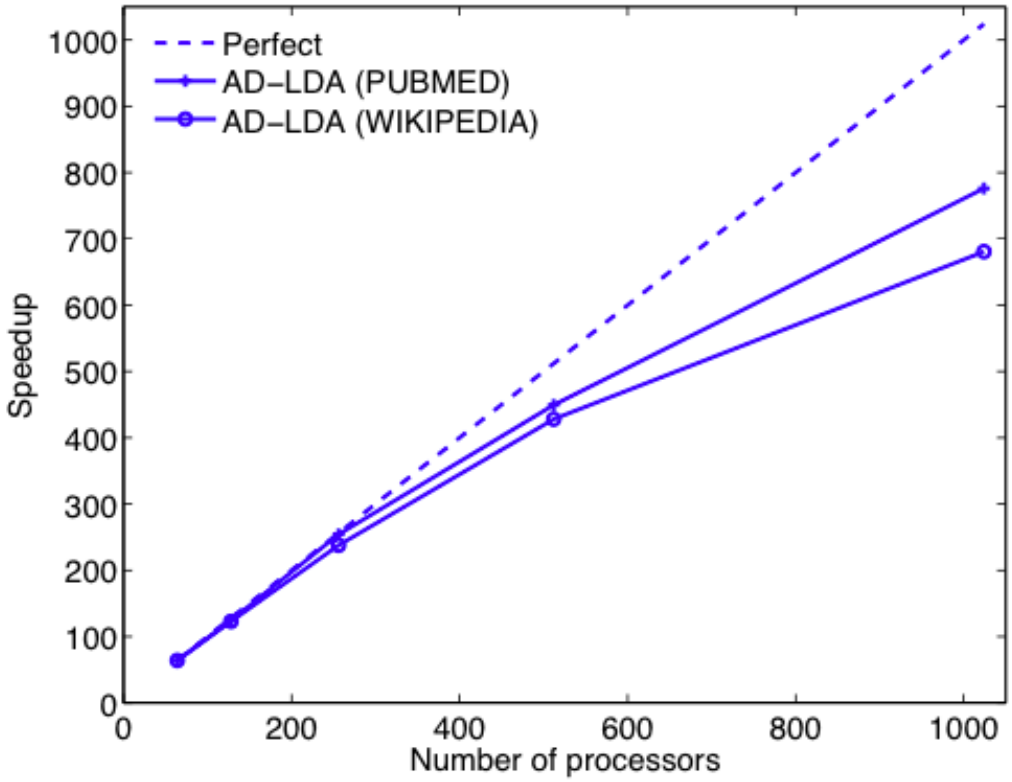



Figure 12: Parallel speedup results for 64 to 1024 processors on multi-million document data sets WIKIPEDIA and PUBMED.

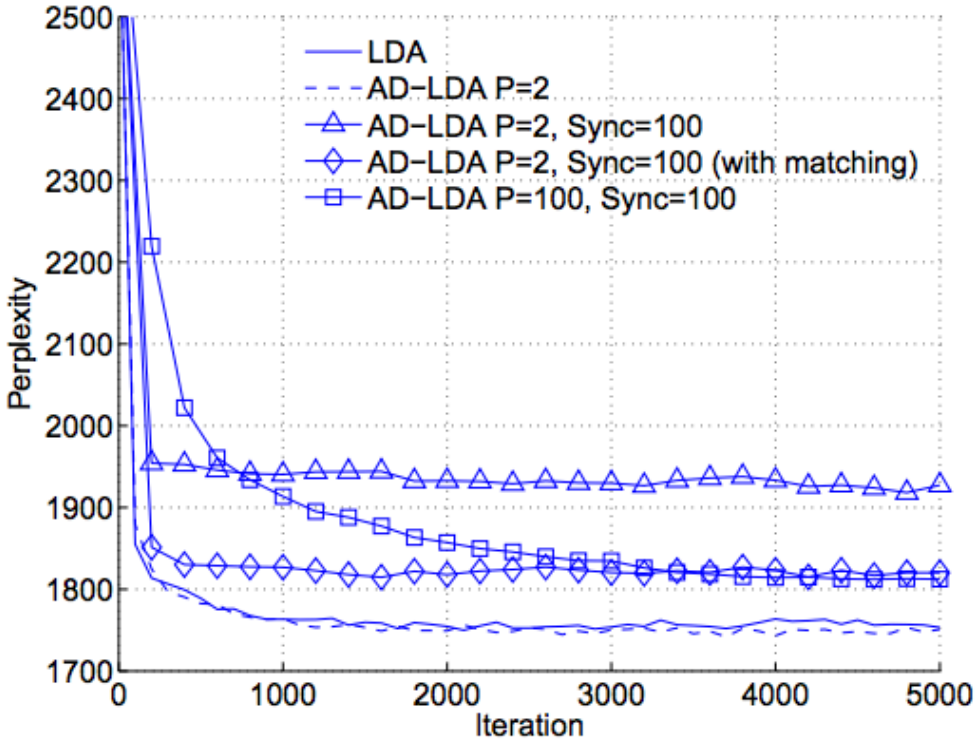



Figure 16: Test perplexity versus iteration where synchronizations between processors only occur every 100 iterations, KOS, $K = 16$.

# Later work….

- Algorithms:
  - Distributed variational EM
  - Asynchronous LDA (AS-LDA)
  - Approximate Distributed LDA (AD-LDA)
  - Ensemble versions of LDA: HLDA, DCM-LDA
- Implementations:
  - GitHub Yahoo_LDA
    - not Hadoop, special-purpose communication code for synchronizing the global counts
    - Alex Smola, Yahoo→CMU
  - Mahout LDA
    - Andy Schlaikjer, CMU→Twitter

# FAST SAMPLING FOR LDA

RECAP

```
    # topic k, docId d, and wordId w are integer indices
    #
    # x[d][j] = w, index of j-th word in doc d
    # z[d][j] = k, index of latent topic of j-th word in doc d
    # vocab[w] = string for the word with index w
    #
    # totalTopicCount[k] = number of words in topic k
    # docTopicCount[d][k] = number of words in topic k for document d
    # wordTopicCount[w][k] = number of occurrences of word w in topic k
    # totalWords = number of words in the corpus
```

linear in corpus size and #topics

time and space

```
def initGibbs(self):
    print '.initializing latent vars'
    self.totalTopicCount = self.topicCounter()
    self.docTopicCount = [self.topicCounter() for d in xrange(len(self.x))]
    self.wordTopicCount = [self.topicCounter() for w in xrange(len(self.vocab))]
    self.z = [[-1 for j in xrange(len(self.x[d]))] for d in xrange(len(self.x))]
    for d in xrange(len(self.x)):
        if (d+1)%self.dstep==0: print '..doc',d+1,'of',len(self.x)
        for j in xrange(len(self.x[d])):
            w = self.x[d][j]
            k = random.randint(0,self.numTopics-1)
            self.z[d][j] = k
            self.docTopicCount[d].add(k, 1)
            self.wordTopicCount[w].add(k, 1)
            self.totalTopicCount.add(k, 1)
    #reasonable parameters
    self.alpha = 1.0/self.numTopics
    self.beta = 1.0/len(self.vocab)
    print "alpha:",self.alpha,"beta:",self.beta
```

RECAP

```
def runGibbs(self,maxT):
    for t in xrange(maxT):
        print '.iteration',t+1,'of',maxT
        for d in xrange(len(self.x)):
            if (d+1)%self.dstep==0: print '..doc',d+1,'of',len(self.x)
            for j in xrange(len(self.x[d])):
                k = self.resample(d,j)
                self.flip(d, j, self.z[d][j], k)
```

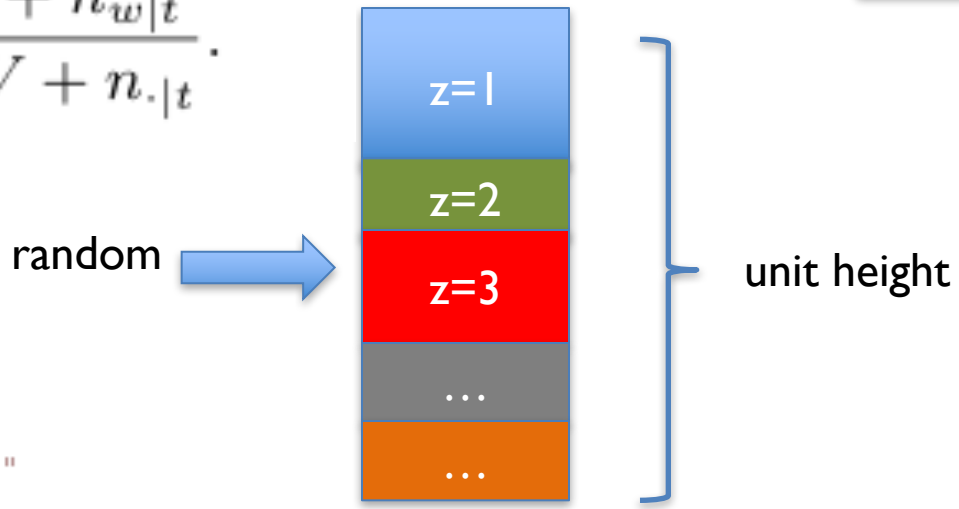
each iteration: linear in corpus size

resample: linear in #topics

```
def resample(self,d,j):
    """sample a new value of z[d][j]"""
    p = []
    norm = 0.0
    #compute pk = Pr(z_dj=k | everything else)
    for k in xrange(self.numTopics):
        w = self.x[d][j]
        z_dj_equals_k = 1 if self.z[d][j]==k else 0
        #unnormalized chance of picking topic k in doc d
        ak = (self.docTopicCount[d][k] - z_dj_equals_k + self.alpha)
        #unnormalized chance of picking topic k for word w
        bk = ((self.wordTopicCount[w][k] - z_dj_equals_k + self.beta)
              /(self.totalTopicCount[k] - z_dj_equals_k + self.totalWords * self.beta))
        pk = ak*bk
        p.append(pk)
        norm += pk
    #pick randomly from the normalized pk
    sum_p_up_to_k = 0.0
    r = random.random()
    for k in xrange(self.numTopics):
        sum_p_up_to_k += p[k]/norm
        if r < sum_p_up_to_k:
            return k
```

most of the time is resampling

$$P(z = t|w) \quad \propto \quad (\alpha_t + n_{t|d}) \frac{\beta + n_{w|t}}{\beta V + n_{\cdot|t}}.$$




```
def resample(self,d,j):
    """sample a new value of z[d][j]"""
    p = []
    norm = 0.0
    #compute pk = Pr(z_dj=k | everything else)
    for k in xrange(self.numTopics):
        w = self.x[d][j]
        z_dj_equals_k = 1 if self.z[d][j]==k else 0
        #unnormalized chance of picking topic k in doc d
        ak = (self.docTopicCount[d][k] - z_dj_equals_k + self.alpha)
        #unnormalized chance of picking topic k for word w
        bk = ((self.wordTopicCount[w][k] - z_dj_equals_k + self.beta)
              /(self.totalTopicCount[k] - z_dj_equals_k + self.totalWords * self.beta))
        pk = ak*bk
        p.append(pk)
        norm += pk
    #pick randomly from the normalized pk
    sum_p_up_to_k = 0.0
    r = random.random()
    for k in xrange(self.numTopics):
        sum_p_up_to_k += p[k]/norm
        if r < sum_p_up_to_k:
            return k
```

1. You spend a *lot* of time sampling
2. There's a loop over all topics here in the sampler

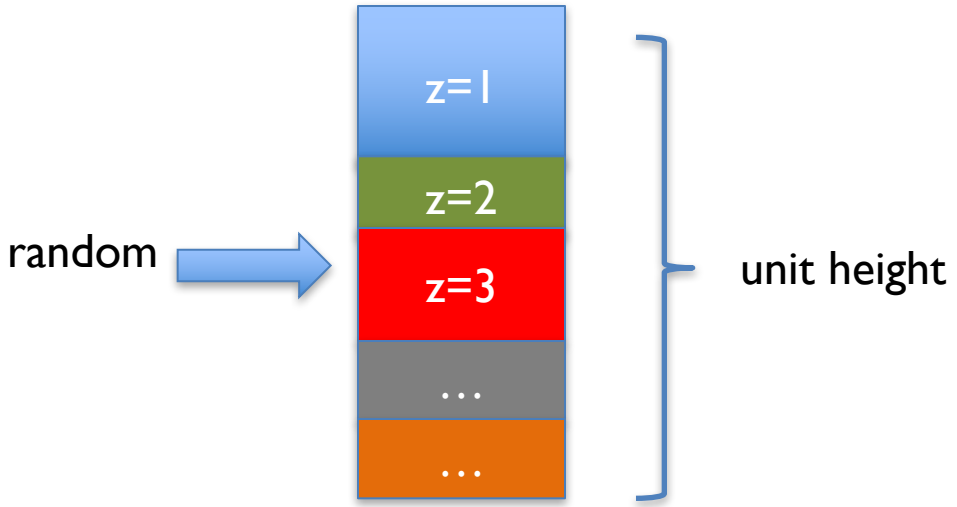
# Efficient Methods for Topic Model Inference on Streaming Document Collections

Limin Yao, David Mimno, and Andrew McCallum
Department of Computer Science
University of Massachusetts, Amherst
{lmyao, mimno, mccallum}@cs.umass.edu

KDD 09

$$P(z = t|w) \quad \propto \quad (\alpha_t + n_{t|d}) \frac{\beta + n_{w|t}}{\beta V + n_{\cdot|t}}.$$

$$P(z = t|w) \propto \frac{\alpha_t \beta}{\beta V + n_{\cdot|t}} + \frac{n_{t|d} \beta}{\beta V + n_{\cdot|t}} + \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{\cdot|t}}.$$

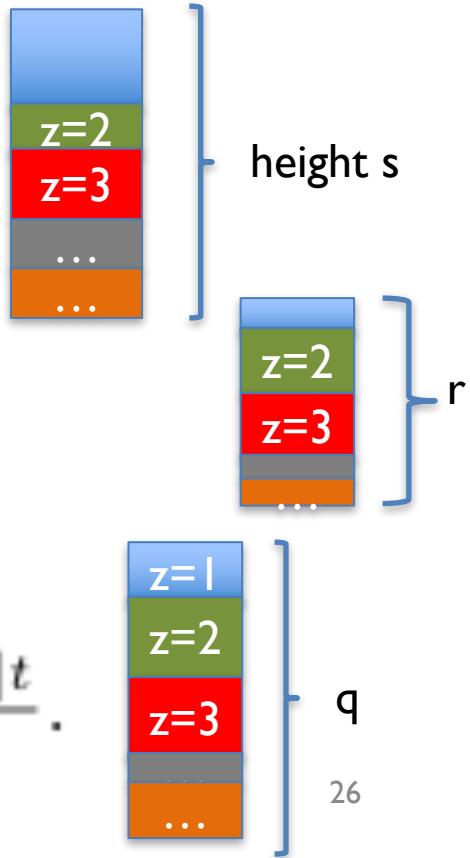


$$P(z = t|w) \quad \propto \quad (\alpha_t + n_{t|d}) \frac{\beta + n_{w|t}}{\beta V + n_{\cdot|t}}.$$

$$P(z = t|w) \propto \frac{\alpha_t \beta}{\beta V + n_{\cdot|t}} + \frac{n_{t|d} \beta}{\beta V + n_{\cdot|t}} + \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{\cdot|t}}.$$

$z=s+r+q$

$$\begin{cases} s = \sum_t \frac{\alpha_t \beta}{\beta V + n_{\cdot|t}} \\ r = \sum_t \frac{n_{t|d} \beta}{\beta V + n_{\cdot|t}} \\ q = \sum_t \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{\cdot|t}}. \end{cases}$$



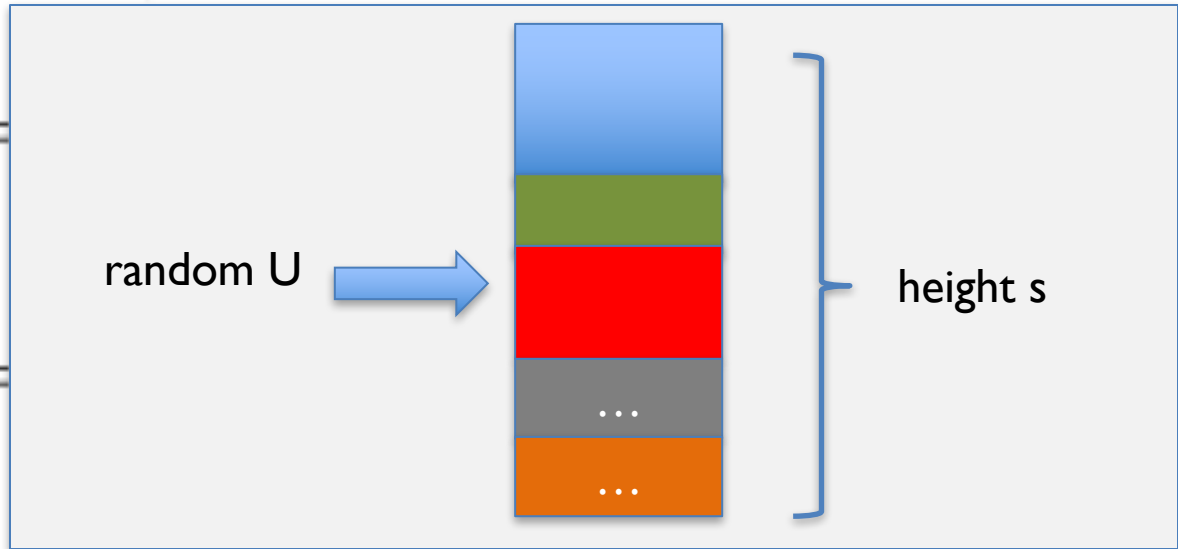
- Draw random U from uniform[0,1]
- If *U*<*s*:
  - lookup *U* on line segment with tic-marks at $\alpha_1\beta/(\beta V + n_{\cdot|1})$, $\alpha_2\beta/(\beta V + n_{\cdot|2})$, ...

*alizer = s+r+q*

$$s = \sum_t \frac{\alpha_t \beta}{\beta V + n_{\cdot|t}}$$

$$r =$$

$$q =$$



- If *U*<*s*:
  - lookup *U* on line segment with tic-marks at $\alpha_1\beta/(\beta V + n_{\cdot|1})$, $\alpha_2\beta/(\beta V + n_{\cdot|2})$, ...
- If *s*<*U*<*r*:
  - lookup *U* on line segment for *r*

Only need to check t such that $n_{t|d}>0$

*z*=*s*+*r*+*q*

$$\begin{cases} s = \sum_t \frac{\alpha_t \beta}{\beta V + n_{\cdot|t}} \\ r = \sum_t \frac{n_{t|d}\beta}{\beta V + n_{\cdot|t}} \\ q = \sum_t \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{\cdot|t}}. \end{cases}$$

- If $U<s$:
  - lookup $U$ on line segment with tic-marks at $\alpha_1\beta/(\beta V + n_{\cdot|1})$, $\alpha_2\beta/(\beta V + n_{\cdot|2})$, ...
- If $s<U<s+r$:
  - lookup $U$ on line segment for $r$
- If $s+r<U$:
  - lookup $U$ on line segment for $q$

$z=s+r+q$

$$r = \sum_t \frac{n_{t|d}\beta}{\beta V + n_{\cdot|t}}$$

$$q = \sum_t \frac{(\alpha_t + n_{t|d})n_{w|t}}{\beta V + n_{\cdot|t}}.$$

Only need to check t such that $n_{w|t}>0$

Only need to check occasionally (< 10% of the time)

Only need to check t such that $n_{t|d}>0$

Only need to check t such that $n_{w|t}>0$

$z=s+r+q$

$$\begin{cases} s = \sum_t \frac{\alpha_t \beta}{\beta V + n_{\cdot|t}} \\ r = \sum_t \frac{n_{t|d} \beta}{\beta V + n_{\cdot|t}} \\ q = \sum_t \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{\cdot|t}}. \end{cases}$$

Only need to **store** (and maintain) total words per topic and α's,β,*V*

Trick; count up $n_{t|d}$ for *d* when you start working on *d* and update incrementally

Only need to **store** $n_{t|d}$ for current *d*

Need to **store** $n_{w|t}$ for each word, topic pair …???

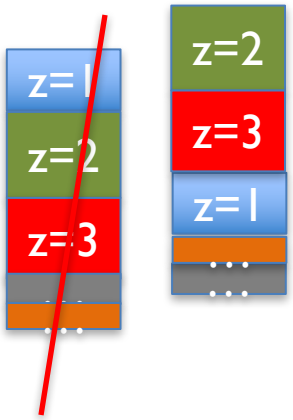
*z*=*s*+*r*+*q*

$$
\begin{aligned}
s &= \sum_t \frac{\alpha_t \beta}{\beta V + n_{\cdot|t}} \\
r &= \sum_t \frac{n_{t|d} \beta}{\beta V + n_{\cdot|t}} \\
q &= \sum_t \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{\cdot|t}}.
\end{aligned}
$$

$$q = \sum_t \left[ \frac{\alpha_t + n_{t|d}}{\beta V + n_{\cdot|t}} \times n_{w|t} \right].$$

*1. Precompute, for each t,* $\frac{\alpha_t + n_{t|d}}{\beta V + n_{\cdot|t}}$

*2. Quickly find t's such that* $n_{w|t}$ *is large for w*




$z=s+r+q$

$$\begin{cases} s = \sum_t \frac{\alpha_t \beta}{\beta V + n_{\cdot|t}} \\ r = \sum_t \frac{n_{t|d} \beta}{\beta V + n_{\cdot|t}} \\ q = \sum_t \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{\cdot|t}}. \end{cases}$$

Most (>90%) of the time and space is here…

Need to **store** $n_{w|t}$ for each word, topic pair …???

$$q = \sum_t \left[ \frac{\alpha_t + n_{t|d}}{\beta V + n_{\cdot|t}} \times n_{w|t} \right].$$

*1. Precompute, for each t,* $\frac{\alpha_t + n_{t|d}}{\beta V + n_{\cdot|t}}$

*2. Quickly find t's such that* $n_{w|t}$ *is large for w*

- associate each w with an int array
  - no larger than frequency of w
  - no larger than #topics
- encode *(t,n)* as a bit vector
  - *n* in the high-order bits
  - *t* in the low-order bits
- keep ints sorted in descending order

Most (>90%) of the time and space is here…

$n_{w|t}$ for each word, topic pair …???

$$q = \sum_t \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{\cdot|t}}.$$

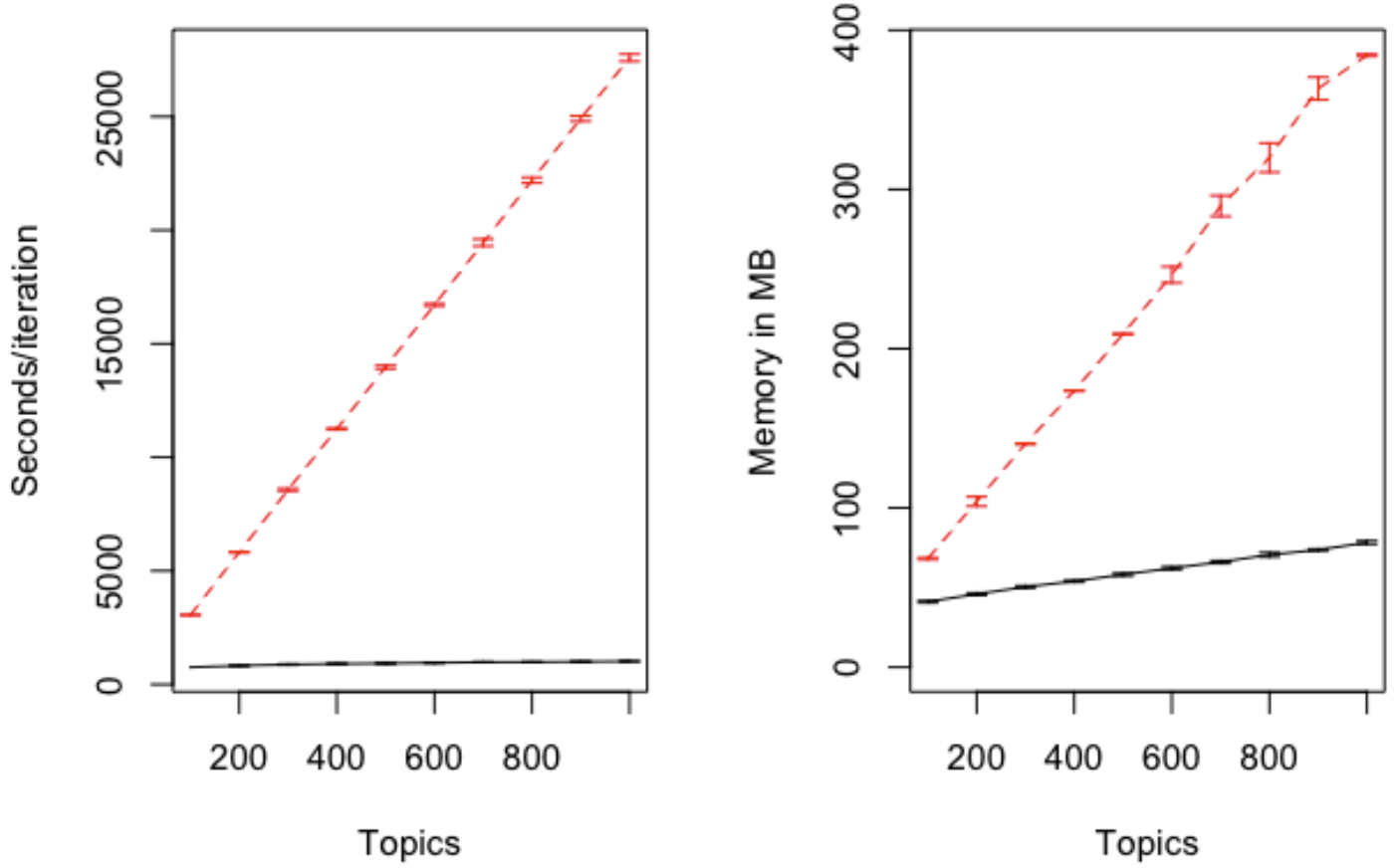


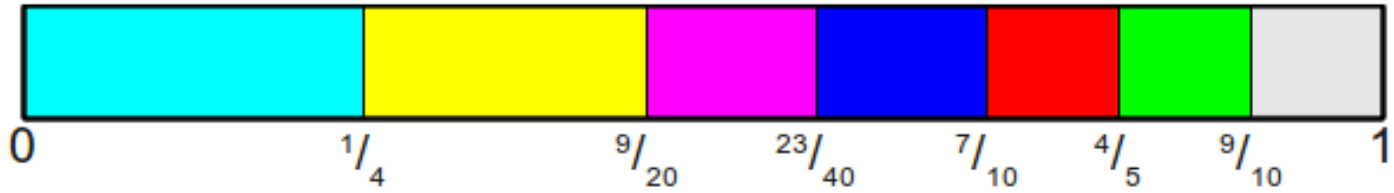

**Figure 2: A comparison of time and space efficiency between standard Gibbs sampling (dashed red lines) and the SparseLDA algorithm and data structure presented in this paper (solid black lines). Error bars show the standard deviation over five runs.**

# Other Fast Samplers for LDA

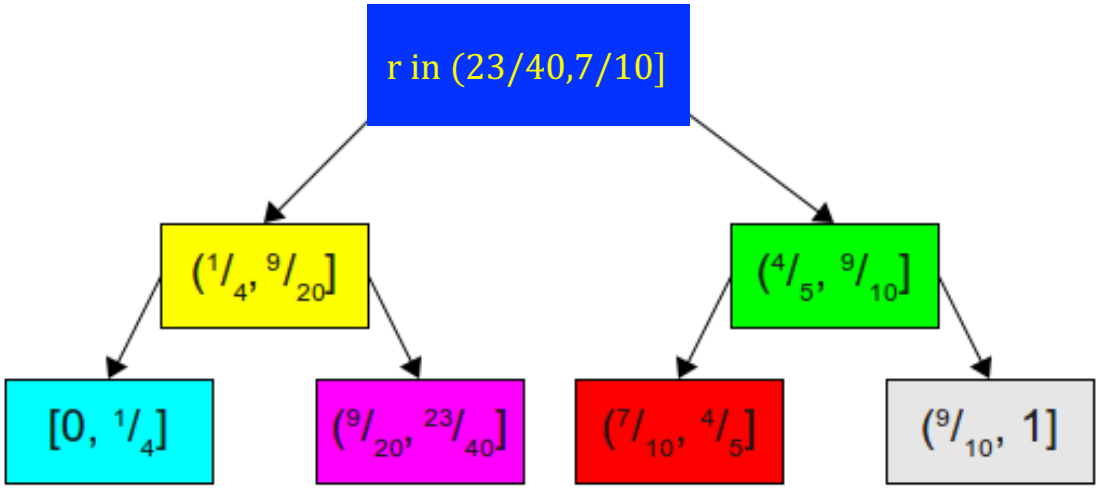
# Alias tables

O(K)

Basic problem: how can we sample from a biased coin quickly?




If the distribution changes slowly maybe we can do some preprocessing and then sample multiple times. Proof of concept: generate *r~uniform* and use a binary tree



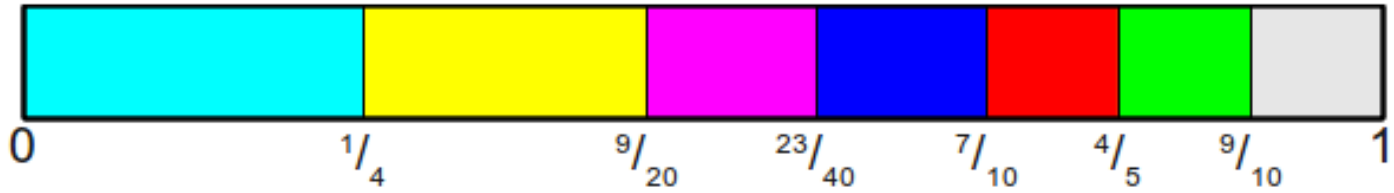

O(log2K)

http://www.keithschwarz.com/darts-dice-coins/

# Alias tables

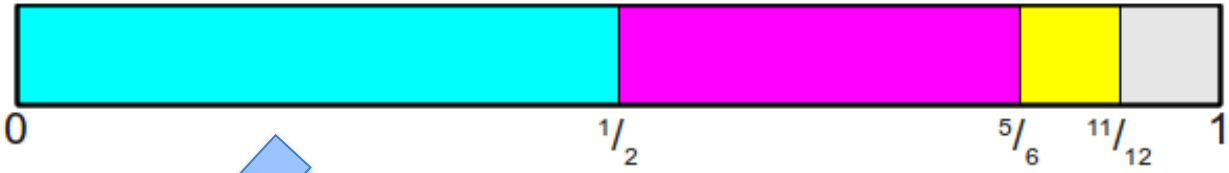
O(K)

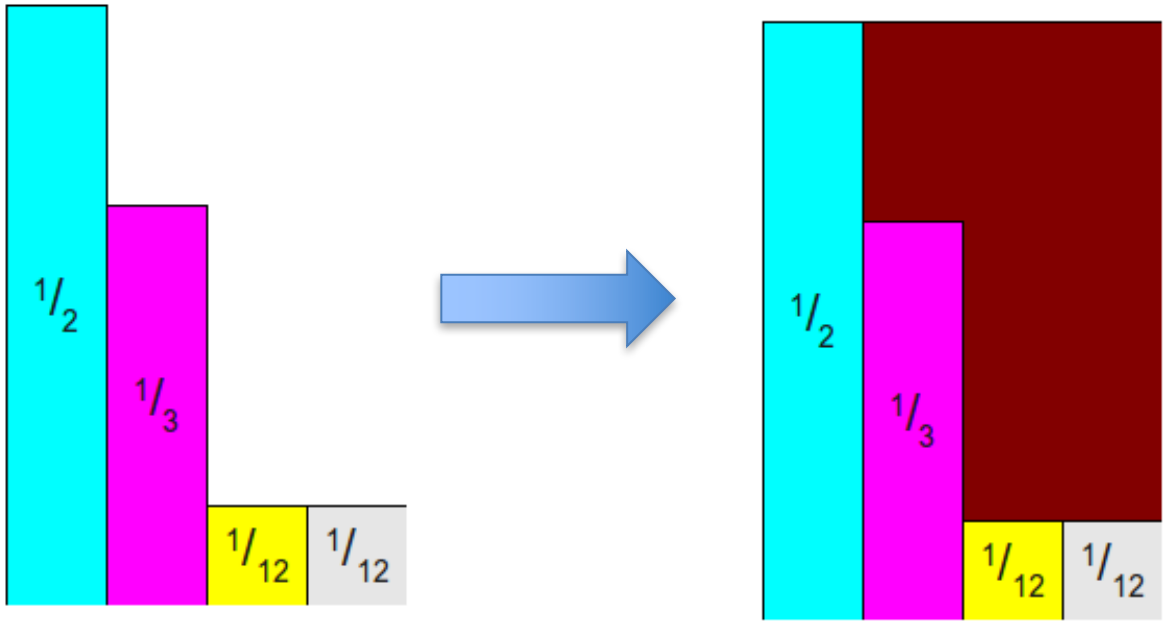
Basic problem: how can we sample from a biased die quickly?



# Alias tables

Another idea…




Simulate the dart with two drawn values:

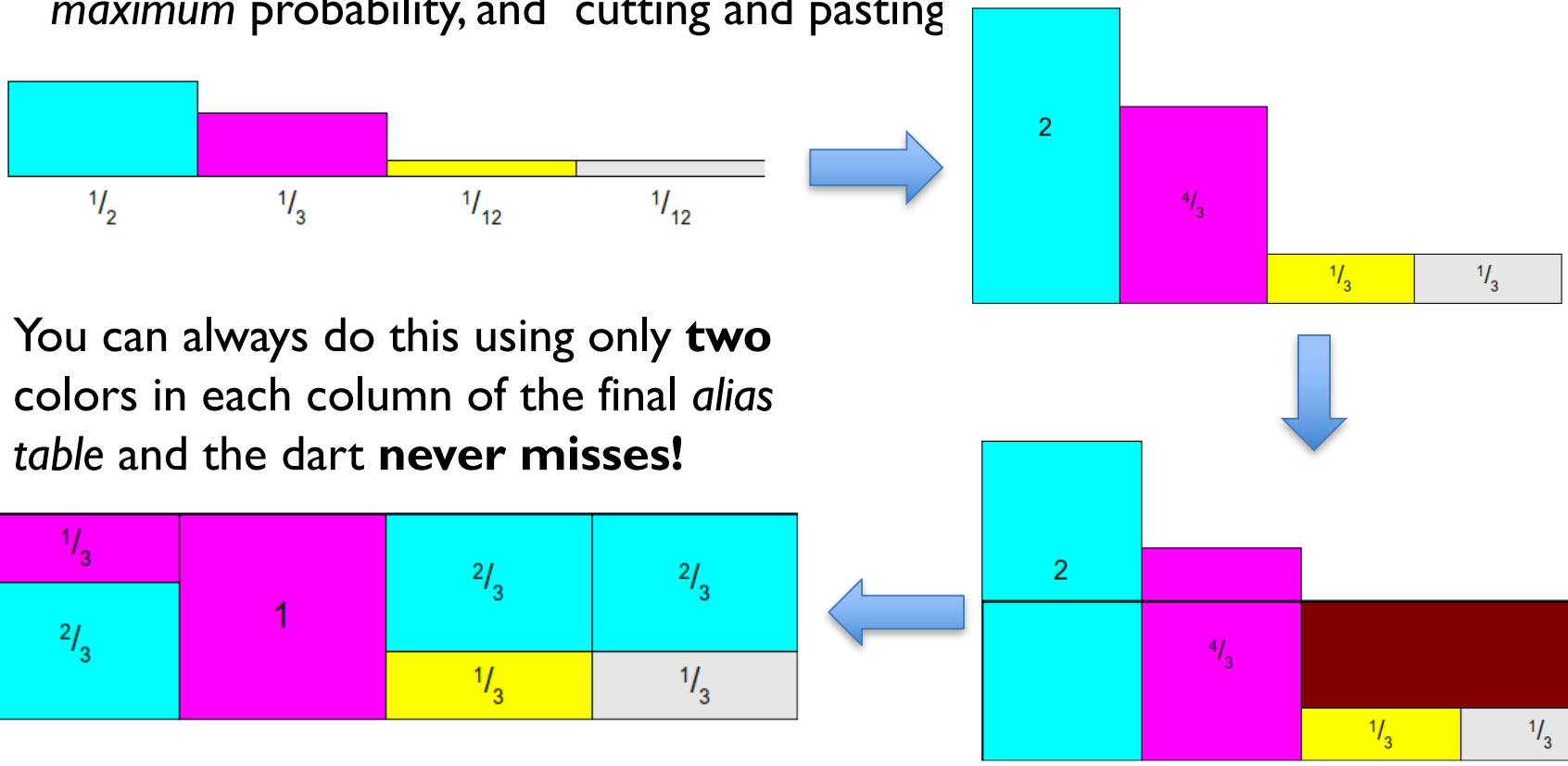
rx ➔ int(u1*K)
ry ➔ u1*$p_{max}$

keep throwing till you hit a stripe

http://www.keithschwarz.com/darts-dice-coins/

# Alias tables

An even more clever idea: minimize the brown space (where the dart "misses") by sizing the rectangle's height to the *average* probability, not the *maximum* probability, and cutting and pasting




You can always do this using only **two** colors in each column of the final *alias table* and the dart **never misses!**

mathematically speaking…

http://www.keithschwarz.com/darts-dice-coins/

# LDA with Alias Sampling

**Reducing the Sampling Complexity of Topic Models**


Aaron Q. Li
CMU Language Technologies
Pittsburgh, PA
aaronli@cmu.edu

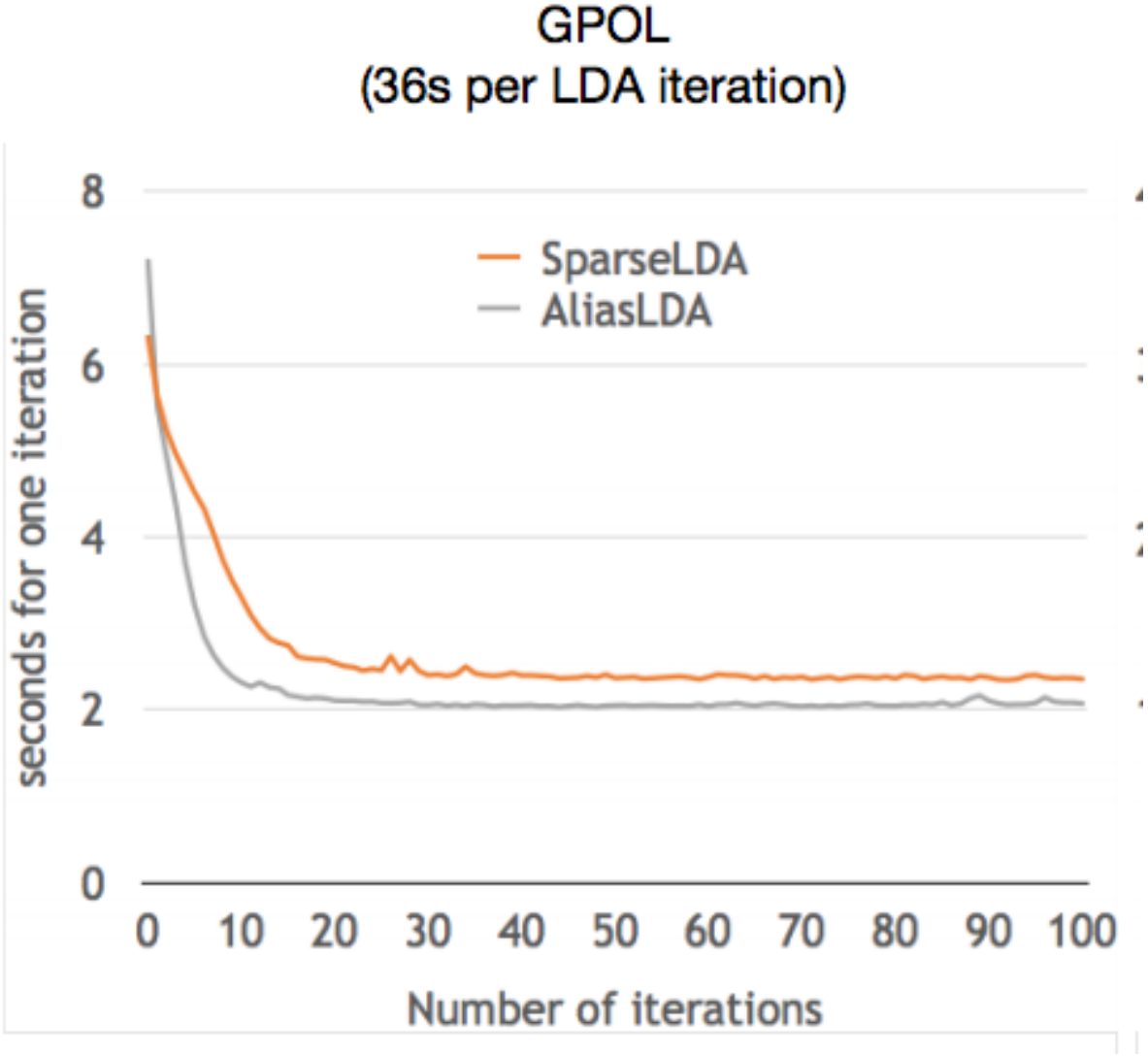
Amr Ahmed
Google Strategic Technologies
Mountain View, CA
amra@google.com

Sujith Ravi
Google Strategic Technologies
Mountain View, CA
sravi@google.com

Alexander J. Smola
CMU MLD and Google ST
Pittsburgh PA
alex@smola.org


[KDD 2014]

- Sample Z's with alias sampler
- Don't update the sampler with each flip:
  - Correct for "staleness" with Metropolis-Hastings algorithm


GPOL
(36s per LDA iteration)
SparseLDA
AliasLDA
seconds for one iteration
8
6
4
2
0
0
10
20
30
40
50
60
70
80
90
100
Number of iterations

# Yet More Fast Samplers for LDA

# A Scalable Asynchronous Distributed Algorithm for Topic Modeling

Hsiang-Fu Yu
Univ. of Texas, Austin
rofuyu@cs.u[Alias tables].edu

Cho-Jui Hsieh
Univ. of Texas, Austin
cjhsieh@cs.utexas.edu

Hyokun Yun
Amazon
yunhyoku@amazon.com

S.V.N Vishwanathan
Univ. of California, Santa Cruz
vishy@ucsc.edu

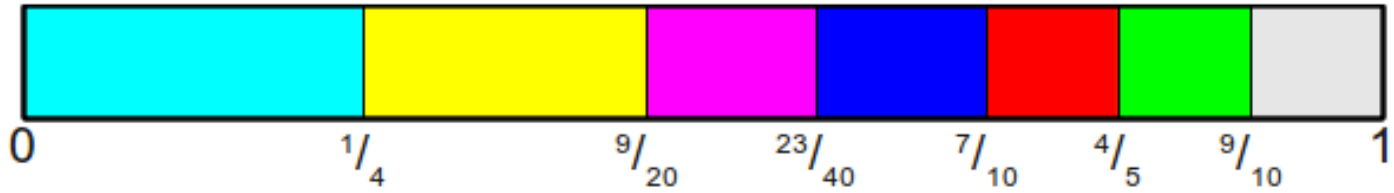
Inderjit S. Dhillon
Univ. of Texas, Austin
inderjit@cs.utexas.edu

WWW 2015

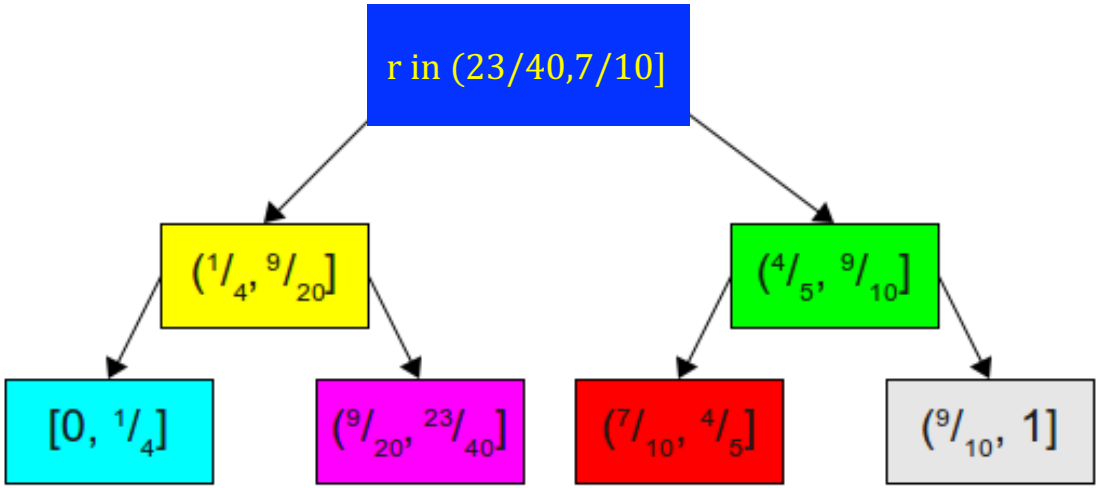
# Fenwick Tree (1994)

O(K)

Basic problem: how can we sample from a biased die quickly….




…and update quickly? maybe we can use a binary tree….



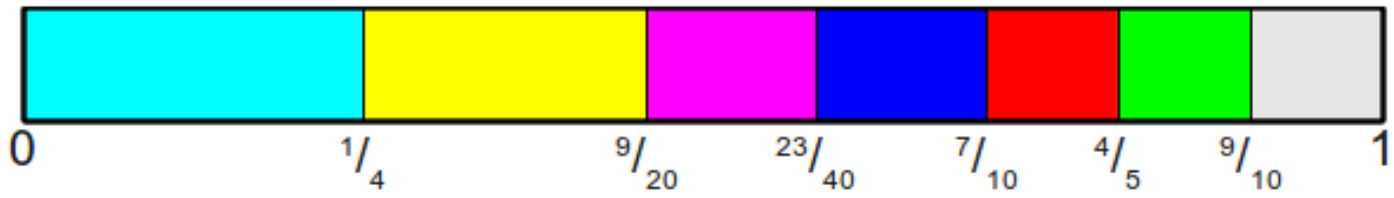

O(log2K)

http://www.keithschwarz.com/darts-dice-coins/

# Data structures and algorithms

| | Data Structure Space | Initialization Time | Initialization Space | Generation Time | Parameter Update Time |
|---|---|---|---|---|---|
| LSearch | $c_T = \mathbf{p}^\top \mathbf{1}$: $O(1)$ | $O(T)$ | $O(1)$ | $O(T)$ | $O(1)$ |
| BSearch | $\mathbf{c} = \mathtt{cumsum}(\mathbf{p})$: $O(T)$ | $O(T)$ | $O(1)$ | $O(\log T)$ | $O(T)$ |
| Alias Method | $prob, alias$: $O(T)$ | $O(T)$ | $O(T)$ | $O(1)$ | $O(T)$ |
| F+tree Sampling | $\mathtt{F.initialize}(\mathbf{p})$: $O(T)$ | $O(T)$ | $O(1)$ | $O(\log T)$ | $O(\log T)$ |

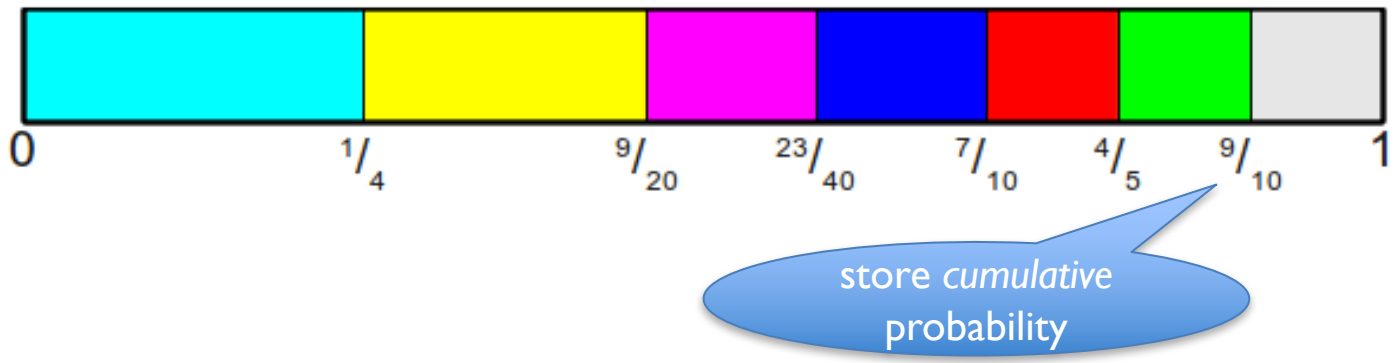
LSearch: linear search



# Data structures and algorithms

| | Data Structure Space | Initialization Time | Initialization Space | Generation Time | Parameter Update Time |
|---|---|---|---|---|---|
| LSearch | $c_T = \mathbf{p}^\top \mathbf{1}$: $O(1)$ | $O(T)$ | $O(1)$ | $O(T)$ | $O(1)$ |
| BSearch | $\mathbf{c} = \mathtt{cumsum}(\mathbf{p})$: $O(T)$ | $O(T)$ | $O(1)$ | $O(\log T)$ | $O(T)$ |
| Alias Method | $prob, alias$: $O(T)$ | $O(T)$ | $O(T)$ | $O(1)$ | $O(T)$ |
| F+tree Sampling | $\mathtt{F.initialize}(\mathbf{p})$: $O(T)$ | $O(T)$ | $O(1)$ | $O(\log T)$ | $O(\log T)$ |

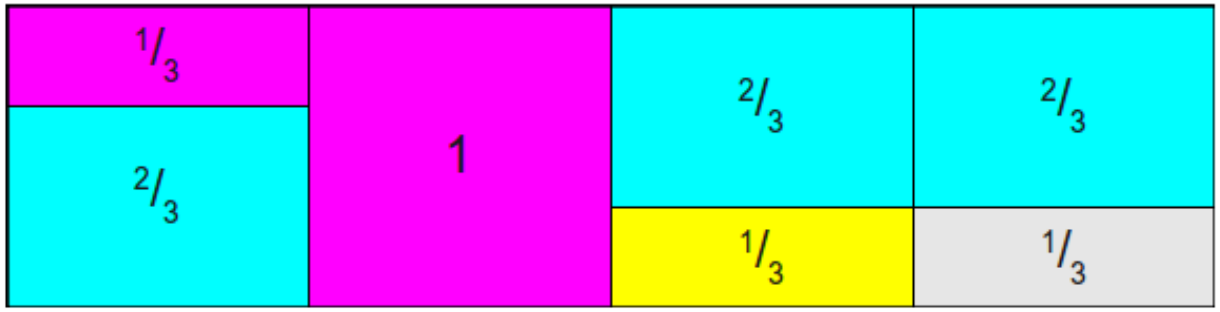
BSearch: binary search



# Data structures and algorithms

| | Data Structure Space | Initialization Time | Initialization Space | Generation Time | Parameter Update Time |
|---|---|---|---|---|---|
| LSearch | $c_T = \mathbf{p}^\top \mathbf{1}$: $O(1)$ | $O(T)$ | $O(1)$ | $O(T)$ | $O(1)$ |
| BSearch | $\mathbf{c} = \mathtt{cumsum}(\mathbf{p})$: $O(T)$ | $O(T)$ | $O(1)$ | $O(\log T)$ | $O(T)$ |
| Alias Method | $prob, alias$: $O(T)$ | $O(T)$ | $O(T)$ | $O(1)$ | $O(T)$ |
| F+tree Sampling | $\mathtt{F.initialize}(\mathbf{p})$: $O(T)$ | $O(T)$ | $O(1)$ | $O(\log T)$ | $O(\log T)$ |

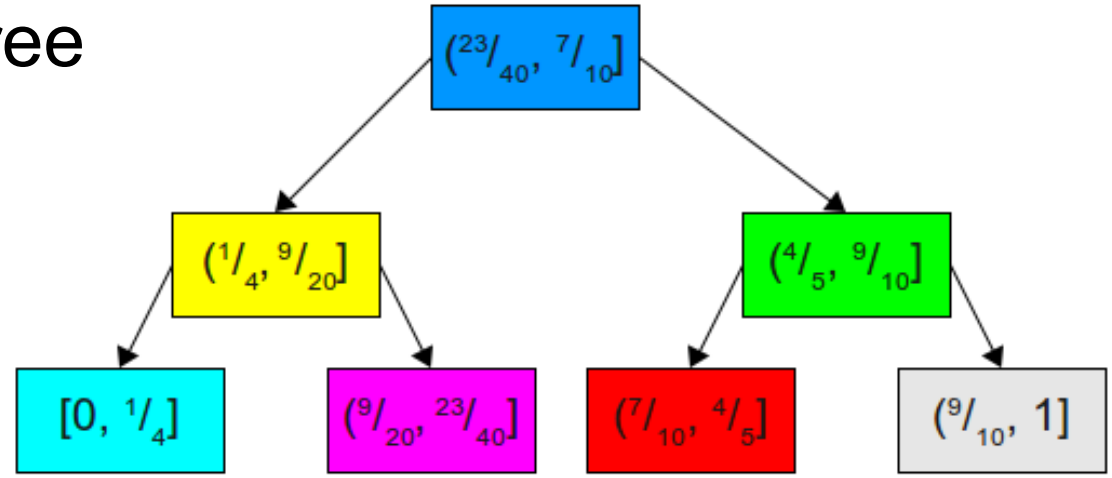
Alias sampling…..



# Data structures and algorithms

| | Data Structure Space | Initialization Time | Initialization Space | Generation Time | Parameter Update Time |
|---|---|---|---|---|---|
| LSearch | $c_T = \mathbf{p}^\top \mathbf{1}$: $O(1)$ | $O(T)$ | $O(1)$ | $O(T)$ | $O(1)$ |
| BSearch | $\mathbf{c} = \mathtt{cumsum}(\mathbf{p})$: $O(T)$ | $O(T)$ | $O(1)$ | $O(\log T)$ | $O(T)$ |
| Alias Method | $prob, alias$: $O(T)$ | $O(T)$ | $O(T)$ | $O(1)$ | $O(T)$ |
| F+tree Sampling | $\mathtt{F.initialize}(\mathbf{p})$: $O(T)$ | $O(T)$ | $O(1)$ | $O(\log T)$ | $O(\log T)$ |

Fenwick tree



# Data structures and algorithms

$$p_t = \frac{(n_{td} + \alpha)(n_{tw} + \beta)}{n_t + \bar{\beta}}, \quad \forall t = 1, \ldots, T.$$

$$= \beta \left( \frac{n_{td} + \alpha}{n_t + \bar{\beta}} \right) + n_{tw} \left( \frac{n_{td} + \alpha}{n_t + \bar{\beta}} \right).$$

Fenwick tree

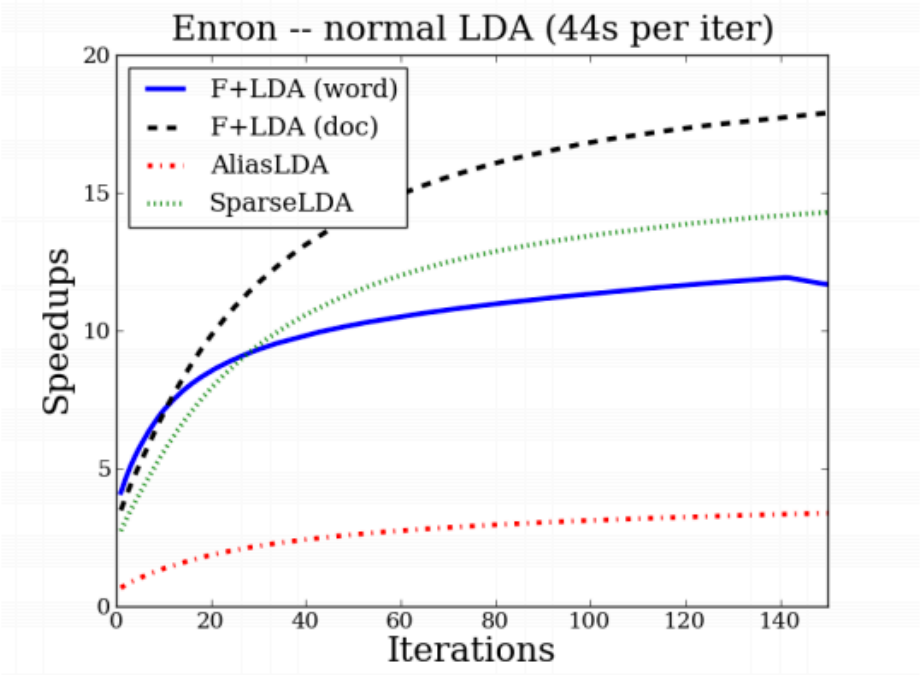
βq: dense, changes slowly, re-used for each word in a document

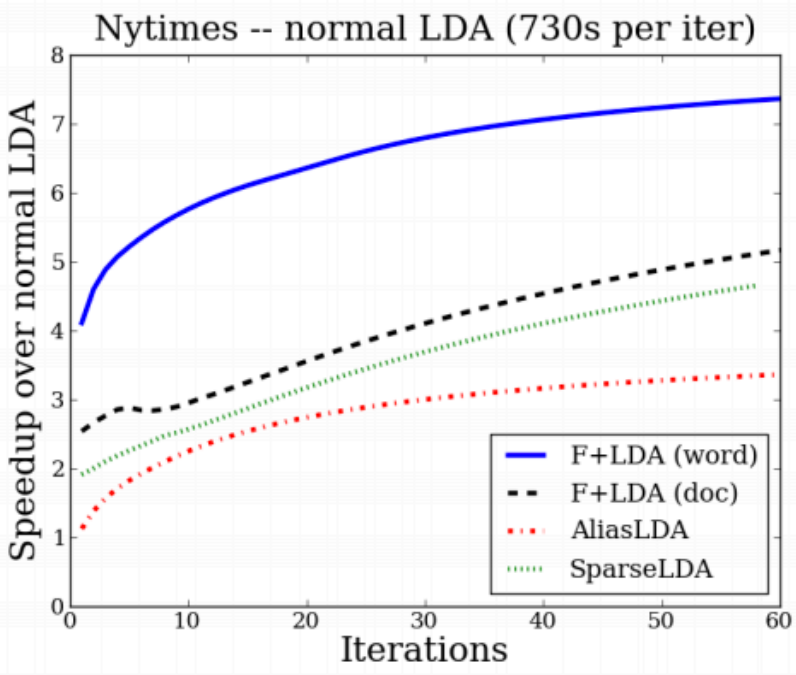
Binary search

r: sparse, a different one is needed for each uniq term in doc

Sampler is:

$$\texttt{discrete}(\mathbf{p}, u) = \begin{cases} \texttt{discrete}(\mathbf{r}, u) & \text{if } u \leq \mathbf{r}^\top \mathbf{1}, \\ \texttt{discrete}(\mathbf{q}, \frac{u - \mathbf{r}^\top \mathbf{1}}{\beta}) & \text{otherwise,} \end{cases}$$

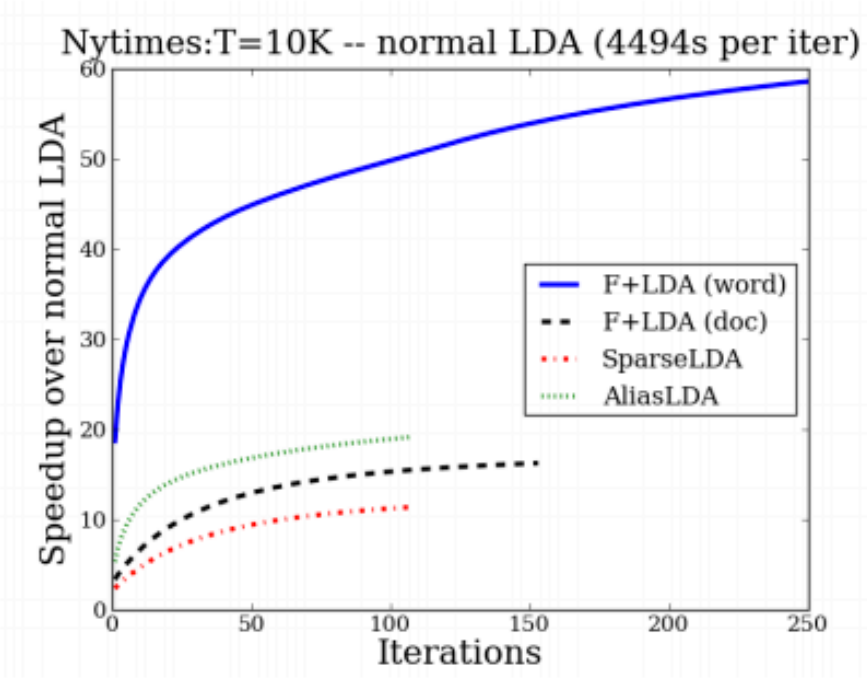


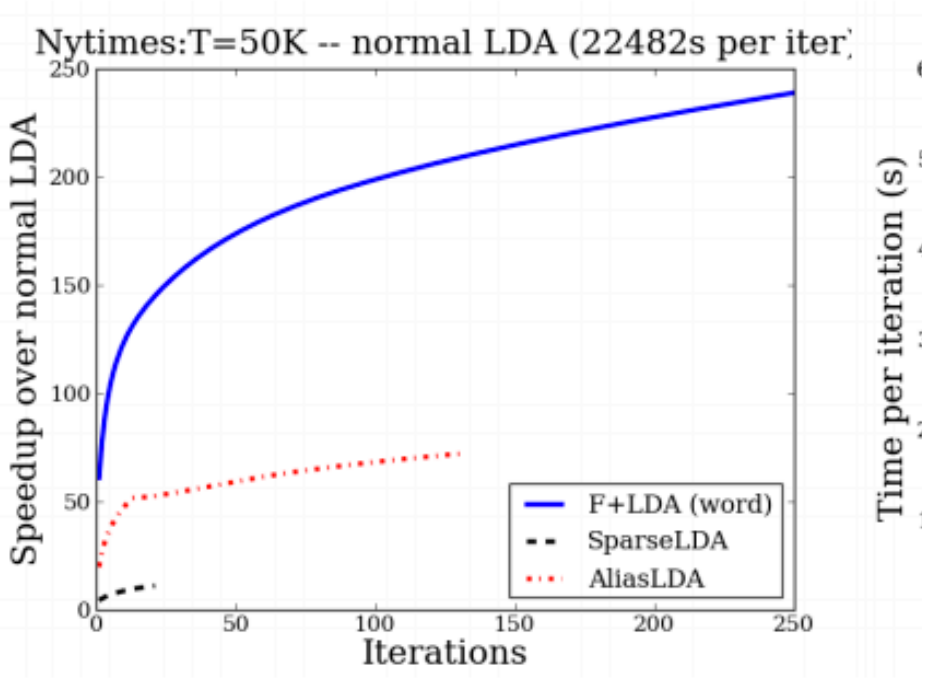




# Speedup vs std LDA sampler (1024 topics)

Table 3: Data statistics.

| | # documents ($I$) | # vocabulary ($J$) | # words |
|---|---|---|---|
| Enron | 37,861 | 28,102 | 6,238,796 |
| NyTimes | 298,000 | 102,660 | 98,793,316 |
| PubMed | 8,200,000 | 141,043 | 737,869,083 |
| Amazon | 29,907,995 | 1,682,527 | 1,499,602,431 |
| UMBC | 40,599,164 | 2,881,476 | 1,483,145,192 |




# Speedup vs std LDA sampler (10k-50k opics)

Table 3: Data statistics.

| | # documents ($I$) | # vocabulary ($J$) | # words |
|---|---|---|---|
| Enron | 37,861 | 28,102 | 6,238,796 |
| NyTimes | 298,000 | 102,660 | 98,793,316 |
| PubMed | 8,200,000 | 141,043 | 737,869,083 |
| Amazon | 29,907,995 | 1,682,527 | 1,499,602,431 |
| UMBC | 40,599,164 | 2,881,476 | 1,483,145,192 |

# And Parallelism….

# A Scalable Asynchronous Distributed Algorithm for Topic Modeling

Hsiang-Fu Yu
Univ. of Texas, Austin
rofuyu@cs.u[Alias tables].edu

Cho-Jui Hsieh
Univ. of Texas, Austin
cjhsieh@cs.utexas.edu

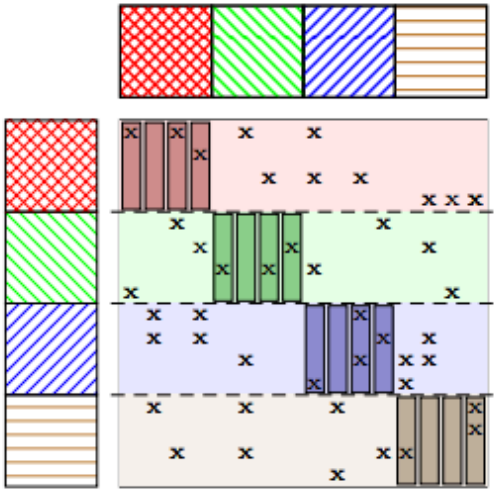
Hyokun Yun
Amazon
yunhyoku@amazon.com

S.V.N Vishwanathan
Univ. of California, Santa Cruz
vishy@ucsc.edu

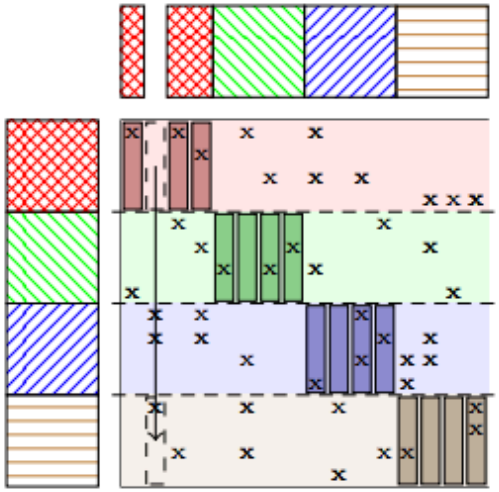
Inderjit S. Dhillon
Univ. of Texas, Austin
inderjit@cs.utexas.edu

Second idea: you can sample document-by-document or word-by-word …. or….

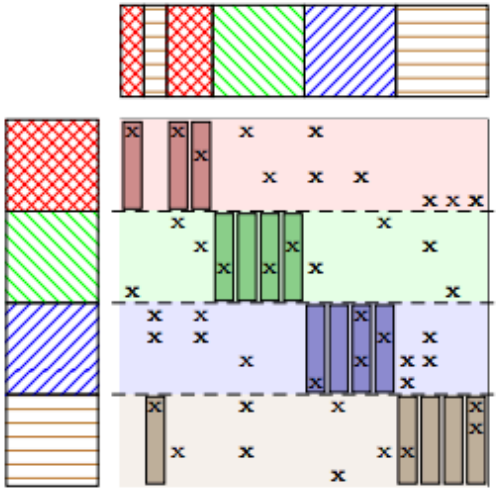
use a MF-like approach to distributing the data.



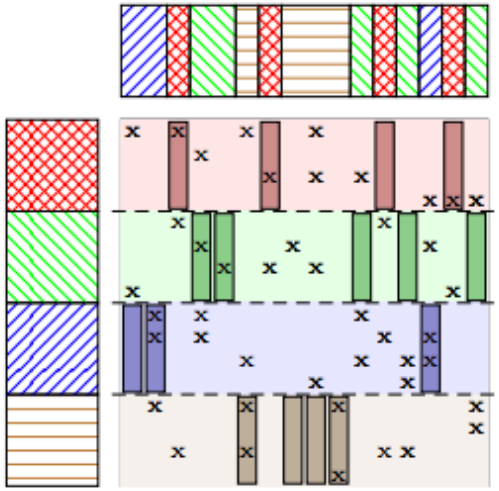
(a) Initial assignment of $\mathbf{w}_j$. Each worker works only on the diagonal active area in the beginning.



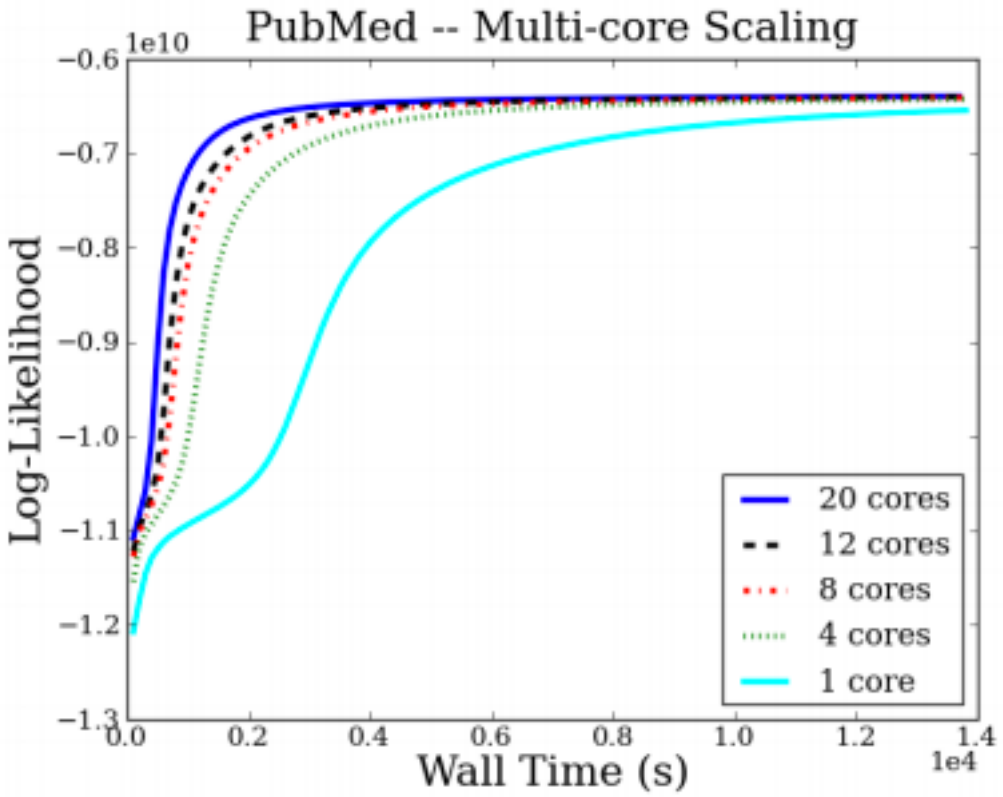
(b) After a worker finishes processing $j$, it sends the corresponding $\mathbf{w}_j$ to another worker. Here, $\mathbf{w}_2$ is sent from worker 1 to 4.



(c) Upon receipt, the $\mathbf{w}_j$ is processed by the new worker. Here, worker 4 can now process $\mathbf{w}_2$ since it owns it.



(d) During the execution of the algorithm, the ownership of the $\mathbf{w}_j$ changes.



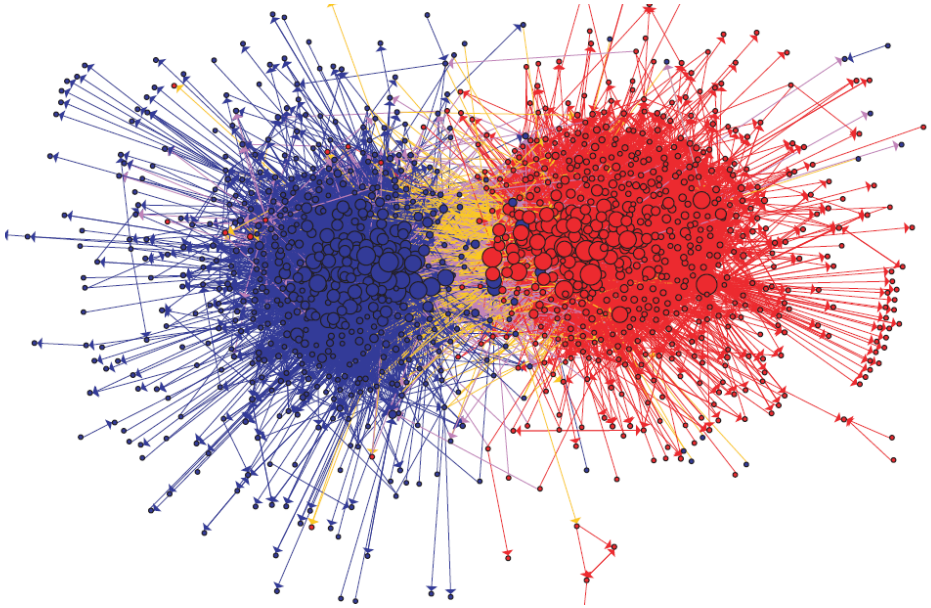

Multi-core NOMAD method

Table 3: Data statistics.

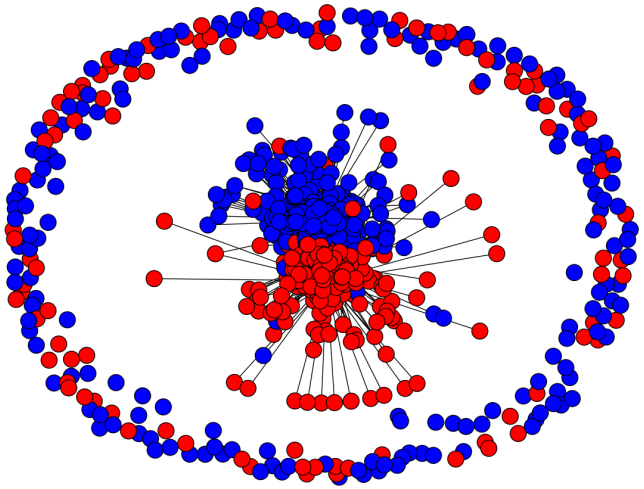
| | # documents ($I$) | # vocabulary ($J$) | # words |
|---|---|---|---|
| Enron | 37,861 | 28,102 | 6,238,796 |
| NyTimes | 298,000 | 102,660 | 98,793,316 |
| PubMed | 8,200,000 | 141,043 | 737,869,083 |

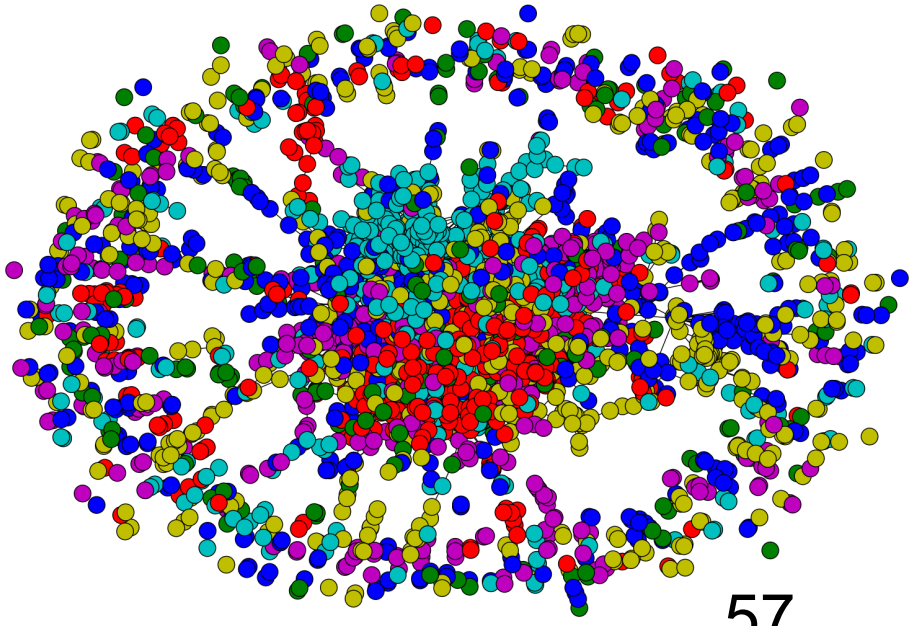
# LDA-LIKE MODELS FOR GRAPHS

# Network Datasets



- UBMCBlog
- AGBlog
- MSPBlog
- Cora
- Citeseer





# Motivation

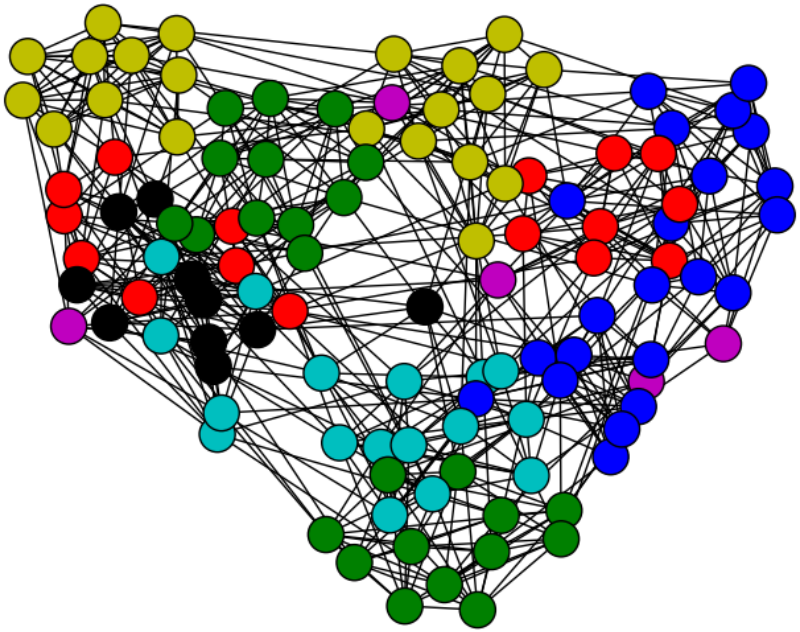
- Social graphs seem to have
  - some aspects of randomness
    - small diameter, giant connected components,..
  - some structure
    - homophily, scale-free degree dist?

- How do you model it?

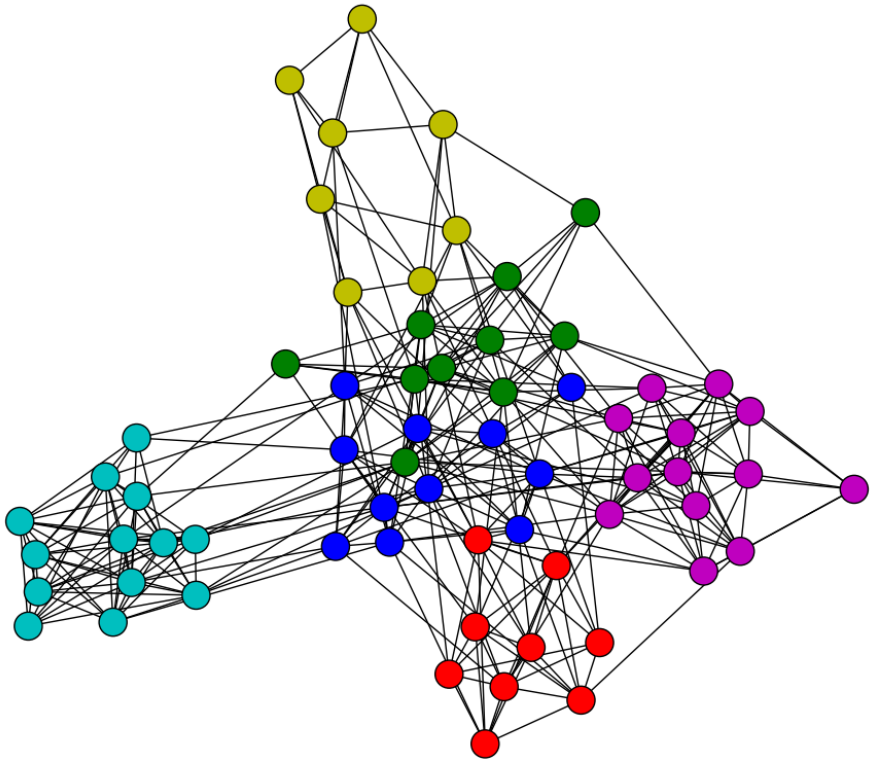
# More terms

- "Stochastic block model", aka "Block-stochastic matrix":
  - Draw $n_i$ nodes in block $i$
  - With probability $p_{ij}$, connect pairs $(u,v)$ where $u$ is in block $i$, $v$ is in block $j$
  - Special, simple case: $p_{ii}=q_i$, and $pij=s$ for all $i \neq j$

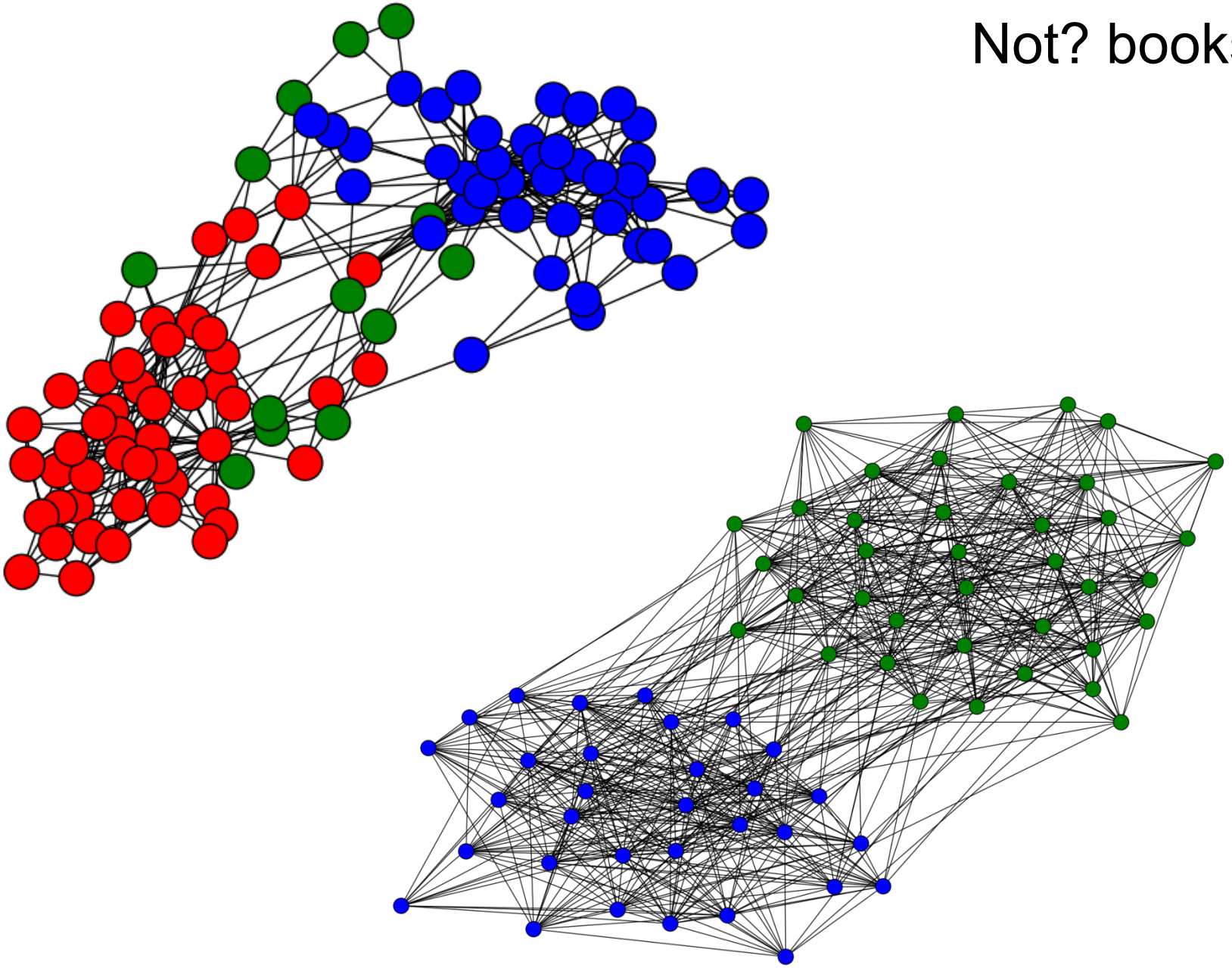
- Question: can you fit this model to a graph?
  - find each $p_{ij}$ and latent node→block mapping



Not? football

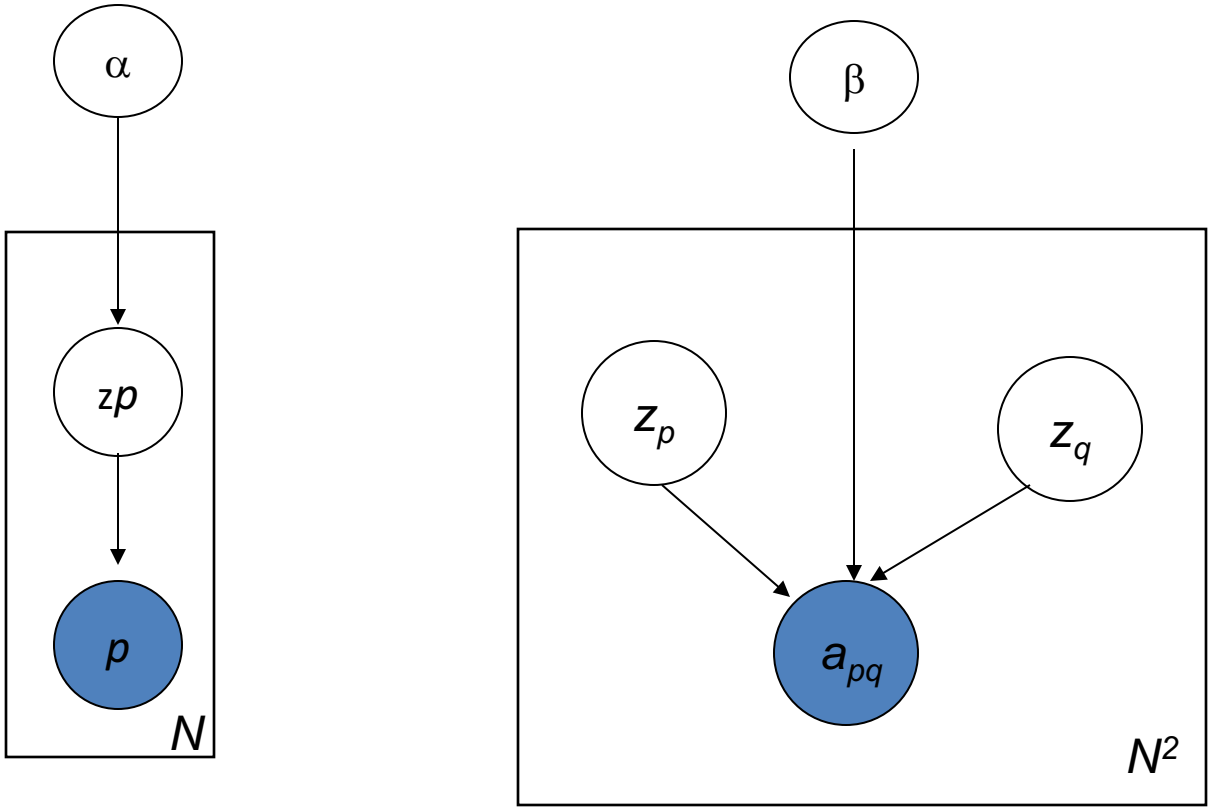


# Not? books

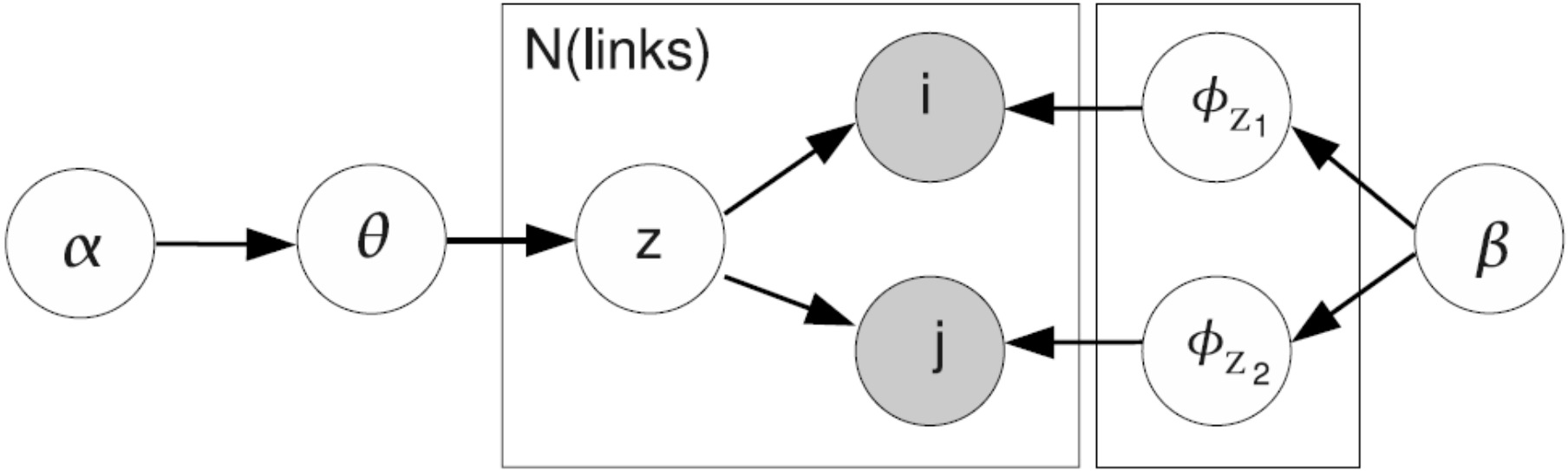


# Stochastic Block models:

assume 1) nodes w/in a block $z$ and
2) edges between blocks $z_p$,$z_q$ are *exchangeable*



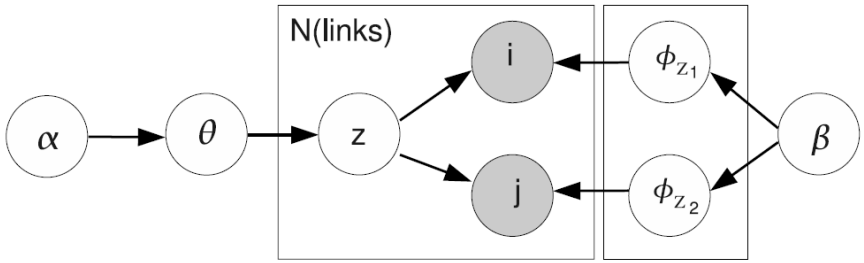
# Another mixed membership block model




$$p(z_l|\{z\}^{\neg l}, \{(i,j)\}^{\neg l}, \alpha, \beta) \propto$$

$$(n_z^{\neg l} + \alpha) \cdot \frac{(q_{z_1 i}^{\neg l} + \beta)(q_{z_2 j}^{\neg l} + \beta)}{(q_{z_1 \cdot}^{\neg l} + M\beta)(q_{z_2 \cdot}^{\neg l} + M\beta + \delta_z)},$$

# Another mixed membership block model




*z=(zi,zj)* is a pair of block ids

$n_z$ = #pairs *z*

$q_{z1,i}$ = #links to *i* from block z1

$q_{z1,\cdot}$ = #outlinks in block z1

δ = indicator for diagonal

M = #nodes

$$p(z_l|\{z\}^{\neg l},\{(i,j)\}^{\neg l},\alpha,\beta) \propto$$

$$(n_z^{\neg l}+\alpha)\cdot\frac{(q_{z_1 i}^{\neg l}+\beta)(q_{z_2 j}^{\neg l}+\beta)}{(q_{z_1\cdot}^{\neg l}+M\beta)(q_{z_2\cdot}^{\neg l}+M\beta+\delta_z)},$$

# Experiments

(e) 1-NN: Social networks

| Dataset | PSK | $PIC_D$ | $PIC_R$ | $PIC_{R4}$ | NCut | NJW |
|---|---|---|---|---|---|---|
| Karate | **1.00** | **1.00** | 0.99 | 0.99 | 1.00 | 0.97 |
| Dolphin | 0.89 | 0.95 | 0.95 | 0.95 | 0.95 | **0.98** |
| UMBC | 0.92 | 0.93 | 0.93 | 0.93 | 0.92 | **0.94** |
| AG | 0.92 | **0.94** | 0.93 | 0.93 | 0.88 | 0.89 |
| MSP | 0.84 | 0.76 | 0.73 | **0.86** | 0.64 | 0.59 |
| Senate | 0.97 | 1.00 | **1.00** | **1.00** | **1.00** | **1.00** |
| PolBook | 0.79 | 0.68 | 0.76 | 0.80 | **0.84** | 0.78 |
| Football | 0.89 | 0.43 | 0.45 | 0.85 | 0.94 | **0.95** |
| MGEmail | 0.22 | 0.27 | 0.26 | 0.72 | 0.80 | **0.81** |
| CiteSeer | 0.34 | 0.55 | 0.54 | **0.71** | 0.69 | 0.66 |
| Cora | 0.45 | 0.56 | 0.51 | **0.80** | 0.47 | 0.75 |
| **Average** | 0.75 | 0.73 | 0.73 | **0.87** | 0.83 | 0.85 |

(f) 1-NN: Author disambiguation

| Dataset | PSK | $PIC_D$ | $PIC_R$ | $PIC_{R4}$ | NCut | NJW |
|---|---|---|---|---|---|---|
| AGupta | 0.68 | 0.74 | 0.72 | **0.95** | 0.79 | 0.91 |
| AKumar | 0.82 | 0.69 | 0.74 | **0.85** | 0.79 | 0.81 |
| CChen | 0.77 | 0.73 | 0.74 | **0.89** | 0.75 | 0.85 |
| DJohnson | 0.81 | 0.81 | 0.83 | **0.95** | 0.85 | 0.92 |
| JLee | 0.55 | 0.61 | 0.68 | **0.92** | 0.79 | 0.91 |
| JMartin | 0.77 | 0.73 | 0.73 | **0.88** | 0.75 | 0.85 |
| JRobinson | 0.86 | 0.75 | 0.80 | **0.92** | 0.83 | 0.85 |
| JSmith | 0.65 | 0.75 | 0.67 | **0.93** | 0.85 | 0.91 |
| KTanaka | 0.81 | 0.84 | 0.86 | **0.95** | 0.90 | 0.90 |
| MBrown | 0.83 | 0.78 | 0.82 | **0.93** | 0.86 | 0.89 |
| MJones | 0.79 | 0.69 | 0.71 | **0.91** | 0.90 | 0.89 |
| MMiller | 0.81 | 0.83 | 0.81 | **0.99** | 0.97 | 0.98 |
| SLee | 0.59 | 0.69 | 0.77 | **0.92** | 0.85 | 0.92 |
| YChen | 0.57 | 0.73 | 0.79 | **0.95** | 0.84 | 0.94 |
| **Average** | 0.74 | 0.74 | 0.76 | **0.92** | 0.84 | 0.90 |

Balasubramanyan, Lin, Cohen, NIPS w/s 2010