Probabilistic models for corpora and graphs
Review: some generative models

• Multinomial Naïve Bayes

- For each document $d = 1, \cdots, M$
  - Generate $C_d \sim \text{Mult}(C | \pi)$
- For each position $n = 1, \cdots, N_d$
  - Generate $w_n \sim \text{Mult}(W | \gamma)$
Review: some generative models

- Multinomial Naïve Bayes

\[ \pi \]

\[ C \]

\[ W_1 \quad W_2 \quad W_3 \quad \cdots \quad W_N \]

\[ \gamma \]

- For each document \( d = 1, \ldots, M \)
  - Generate \( C_d \sim \text{Mult}(C | \pi) \)
  - For each position \( n = 1, \ldots, N_d \)
    - Generate \( w_n \sim \text{Mult}(W | \gamma[C_d]) \)
Review: some generative models

- **Multinomial Naïve Bayes**

  ![Diagram of Multinomial Naïve Bayes Model]

  - For each class $k=1,…,K$
    - Generate $\gamma[k] \sim …$
  - For each document $d = 1,\ldots, M$
    - Generate $C_d \sim \text{Mult}( C | \pi)$
  - For each position $n = 1,\ldots, N_d$
    - Generate $w_n \sim \text{Mult}(W | \gamma[C_d])$
Review: some generative models

- Multinomial Naïve Bayes

- For each class $k=1,\ldots,K$
  - Generate $\gamma[k] \sim \text{Dir}(\beta)$

Dirichlet is a prior for multinomials, defined by $k$ params $(\beta_1,\ldots, \beta_K)$ and $\beta_0=\beta_1+\ldots+\beta_K$

MAP for $P(\gamma=i|\beta) = (n_k + \beta_i)/(n + \beta_0)$

Symmetric Dirichlet: all $\beta_i$‘s are equal
Review – unsupervised Naïve Bayes

• Mixture model: unsupervised naïve Bayes model

\[ \pi \]

\[ Z \]

\[ W \]

\[ N \]

\[ M \]

\[ \beta \]

\[ \gamma \]

• For each class \( k = 1, \ldots, K \)
  - Generate \( \gamma[k] \sim \text{Dir}(\beta) \)

• For each document \( d = 1, \ldots, M \)
  - Generate \( C_d \sim \text{Mult}(C | \pi) \)

• For each position \( n = 1, \ldots, N_d \)
  - Generate \( w_n \sim \text{Mult}(W | \gamma[C_d]) \)

Same generative story – different learning problem...
• Latent Dirichlet Allocation

```
For each document \( d = 1, \ldots, M \)
  • Generate \( \theta_d \sim \text{Dir}(\alpha) \)

For each position \( n = 1, \ldots, N_d \)
  • generate \( z_n \sim \text{Mult}(\theta_d) \)
  • generate \( w_n \sim \text{Mult}(\gamma_{z_n}) \)
```

“Mixed membership”
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.
Review - LDA

- **Gibbs sampling**
  - 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

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Initialise $x_{0,1:n}$.</td>
</tr>
<tr>
<td>2.</td>
<td>For $i = 0$ to $N - 1$</td>
</tr>
<tr>
<td></td>
<td>- Sample $x_1^{(i+1)} \sim p(x_1</td>
</tr>
<tr>
<td></td>
<td>- Sample $x_2^{(i+1)} \sim p(x_2</td>
</tr>
<tr>
<td></td>
<td>\vdots</td>
</tr>
<tr>
<td></td>
<td>- Sample $x_j^{(i+1)} \sim p(x_j</td>
</tr>
<tr>
<td></td>
<td>\vdots</td>
</tr>
<tr>
<td></td>
<td>- Sample $x_n^{(i+1)} \sim p(x_n</td>
</tr>
</tbody>
</table>

**Key capability:** estimate distribution of **one** latent variables given the **other latent variables** and observed variables.
D1: “Happy Thanksgiving!”
D2: “Thanksgiving turkey”
D3: “Istambul, Turkey

K=2
D1: “Happy Thanksgiving!”
D2: “Thanksgiving turkey”
D3: “Istambul, Turkey

K=2
D1: “Happy Thanksgiving!”
D2: “Thanksgiving turkey”
D3: “Istambul, Turkey

K=2
Step 1: initialize z’s randomly

D1: “Happy Thanksgiving!”
D2: “Thanksgiving turkey”
D3: “Istambul, Turkey

K=2

\[\alpha\]

\[\theta_1\]

\[\theta_2\]

\[\theta_3\]

1

2

2

1

1

1

happy

thank

thank

turkey

istam

turkey

\[\gamma\]

K=2

\[\beta\]
D1: “Happy Thanksgiving!”
D2: “Thanksgiving turkey”
D3: “Istambul, Turkey

K=2

Step 1: initialize z’s randomly

Step 2: sample $z_{11}$ from:

$$P(z = t | w) \propto \left( \alpha_t + n_{t|d} \right) \frac{\beta + n_{w|t}}{\beta V + n_{.t}}.$$
D1: “Happy Thanksgiving!”
D2: “Thanksgiving turkey”
D3: “Istambul, Turkey

K=2

\[ P(z = t|w) \propto (\alpha_t + n_{t|d}) \frac{\beta + n_{w|t}}{\beta V + n_{.|t}} \]
D1: “Happy Thanksgiving!”
D2: “Thanksgiving turkey”
D3: “Istambul, Turkey

K=2

#times topic t in D1
--------------------
#words in D1
(smoothed by $\alpha$)

$$P(z = t|w) \propto (\alpha_t + n_{t|d}) \frac{\beta + n_{w|t}}{\beta V + n_{.|t}}.$$
D1: “Happy Thanksgiving!”
D2: “Thanksgiving turkey”
D3: “Istambul, Turkey

K = 2

Step 1: initialize z’s randomly
Step 2: sample z_{11} from:
\[ \Pr(z_{11} = k | z_{12}, \ldots, z_{32}, \alpha, \beta) \]
Step 3: sample z_{12}
□ initialisation
zero all count variables, \( n_{m}^{(k)} \), \( n_{m}^{(i)} \), \( n_{k}^{(i)} \), \( n_{k} \)

for all documents \( m \in [1, M] \) do
  for all words \( n \in [1, N_{m}] \) in document \( m \) do
    sample topic index \( z_{m,n} = k \sim \text{Mult}(1/K) \)
    increment document–topic count: \( n_{m}^{(k)} \) + 1
    increment document–topic sum: \( n_{m} \) + 1
    increment topic–term count: \( n_{k}^{(i)} \) + 1
    increment topic–term sum: \( n_{k} \) + 1
  end for
end for

□ Gibbs sampling over burn-in period and sampling period
while not finished do
  for all documents \( m \in [1, M] \) do
    for all words \( n \in [1, N_{m}] \) in document \( m \) do
      □ for the current assignment of \( k \) to a term \( t \) for word \( w_{m,n} \):
      decrement counts and sums: \( n_{m}^{(k)} \) - 1; \( n_{m} \) - 1; \( n_{k}^{(i)} \) - 1; \( n_{k} \) - 1

      sample topic index \( \tilde{k} \sim p(z_{i}|\tilde{z}_{-i}, \tilde{w}) \)

      □ use the new assignment of \( z_{m,n} \) to the term \( t \) for word \( w_{m,n} \) to:
      increment counts and sums: \( n_{m}^{(k)} \) + 1; \( n_{m} \) + 1; \( n_{k}^{(i)} \) + 1; \( n_{k} \) + 1
    end for
  end for
end for

□ check convergence and read out parameters
Review – unsupervised Naïve Bayes

- Mixture model: unsupervised naïve Bayes model

\[ \pi \]

\[ Z \]

\[ W \]

\[ \beta \]

\[ \gamma \]

\[ K \]

\[ M \]

\[ N \]

For each class \( k = 1, \ldots, K \)
- Generate \( \gamma[k] \sim \text{Dir}(\beta) \)

For each document \( d = 1, \ldots, M \)
- Generate \( C_d \sim \text{Mult}(C | \pi) \)

For each position \( n = 1, \ldots, N_d \)
- Generate \( w_n \sim \text{Mult}(W | \gamma[C_d]) \)

Question: could you use collapsed Gibbs sampling here? what would \( \Pr(z_i | z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_m) \) look like?
Models for corpora and graphs
Stochastic Block models:
assume 1) nodes w/in a block $z$ and
2) edges between blocks $z_p,z_q$ are exchangeable

For each node $i$, pick a latent class $z_i$

For each node pair $i,j$, pick an edge weight $a$ by
$\ a \sim \Pr(a \mid \beta[z_i,z_j])$
Another mixed membership block model

- pick a multinomial $\theta$ over pairs of node topics $k_1, k_2$
- for each node topic $k$, pick a multinomial $\phi_k$ over node id’s
- for each edge $e$:
  - pick $z=(k_1, k_2)$
  - pick $i \sim \Pr(. \mid \phi_{k1})$
  - pick $i \sim \Pr(. \mid \phi_{k2})$
Speeding up modeling for corpora and graphs
Solutions

• Parallelize
  – IPM (Newman et al, AD-LDA)
  – Parameter server (Qiao et al, HW7)

• Speedups:
  – Exploit sparsity
  – Fast sampling techniques

• State of the art methods use both....
This is almost all the run-time, and it’s linear in the number of topics….and bigger corpora need more topics…. random

```
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
```

\[
P(z = t|w) \propto (\alpha_t + n_t|d) \frac{\beta + n_w|t}{\beta V + n.|t}.
\]
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
{limyao, mimno, mccallum}@cs.umass.edu

KDD 09
\[ P(z = t | w) \propto (\alpha_t + n_{t|d}) \frac{\beta + n_w|t}{\beta V + n_{.|t}}. \]

\[ P(z = t | w) \propto \frac{\alpha_t \beta}{\beta V + n_{.|t}} + \frac{n_{t|d} \beta}{\beta V + n_{.|t}} + \frac{(\alpha_t + n_{t|d}) n_w|t}{\beta V + n_{.|t}}. \]
\[ P(z = t|w) \propto (\alpha_t + n_{t|d}) \frac{\beta + n_w|t}{\beta V + n.|t} \]

\[ P(z = t|w) \propto \frac{\alpha_t \beta}{\beta V + n.|t} + \frac{n_{t|d} \beta}{\beta V + n.|t} + \frac{(\alpha_t + n_{t|d}) n_w|t}{\beta V + n.|t} \]

\[
\begin{align*}
  s &= \sum_t \frac{\alpha_t \beta}{\beta V + n.|t} \\
  r &= \sum_t \frac{n_{t|d} \beta}{\beta V + n.|t} \\
  q &= \sum_t \frac{(\alpha_t + n_{t|d}) n_w|t}{\beta V + n.|t}
\end{align*}
\]

\[ z = s + r + q \]
\[ P(z = t|w) \propto \frac{\alpha_t \beta}{\beta V + n.|_t} + \frac{n_{t|d} \beta}{\beta V + n.|_t} + \frac{(\alpha_t + n_{t|d})n_w|_t}{\beta V + n.|_t} \]

- If \( U < s \):
  - lookup \( U \) on line segment with tic-marks at \( \alpha_1 \beta/(\beta V + n.|_1), \alpha_2 \beta/(\beta V + n.|_2), \ldots \)
- If \( s < U < r \):
  - lookup \( U \) on line segment for \( r \)

\[
\begin{align*}
s &= \sum_t \frac{\alpha_t \beta}{\beta V + n.|_t} \\
r &= \sum_t \frac{n_{t|d} \beta}{\beta V + n.|_t} \\
q &= \sum_t \frac{(\alpha_t + n_{t|d})n_w|_t}{\beta V + n.|_t}
\end{align*}
\]

\( z = s + r + q \)
\[ P(z = t | w) \propto \frac{\alpha_t \beta}{\beta V + n_{.t}} + \frac{n_{t|d} \beta}{\beta V + n_{.t}} + \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{.t}}. \]

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

\[ r = \sum_{t} \frac{n_{t|d} \beta}{\beta V + n_{.t}} \]
\[ q = \sum_{t} \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{.t}}. \]

\( z = s + r + q \)

Only need to check \( t \) such that \( n_{w|t} > 0 \)
\[ P(z = t|w) \propto \frac{\alpha_t \beta}{\beta V + n.|t} + \frac{n_{t|d} \beta}{\beta V + n.|t} + \frac{(\alpha_t + n_{t|d})n_{w|t}}{\beta V + n.|t}. \]

\[
\begin{align*}
  z &= s + r + q \\
  s &= \sum_t \frac{\alpha_t \beta}{\beta V + n.|t} \\
  r &= \sum_t \frac{n_{t|d} \beta}{\beta V + n.|t} \\
  q &= \sum_t \frac{(\alpha_t + n_{t|d})n_{w|t}}{\beta V + n.|t}.
\end{align*}
\]

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 \( n_{w|t} > 0 \)
\[ P(z = t|w) \propto \frac{\alpha_t \beta}{\beta V + n_{|t}} + \frac{n_{t|d} \beta}{\beta V + n_{|t}} + \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{|t}}. \]

Only need to **store** (and maintain) total words per topic and \( \alpha \)’s, \( \beta \), \( V \)

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

\[ s = \sum_t \frac{\alpha_t \beta}{\beta V + n_{|t}} \]

\[ r = \sum_t \frac{n_{t|d} \beta}{\beta V + n_{|t}} \]

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

\( z = s + r + q \)

Only need to **store** \( n_{t|d} \) for current \( d \)

Need to **store** \( n_{w|t} \) for each word, topic pair...

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

1. Precompute, for each \( t \),
\[ \frac{\alpha_t + n_{t|d}}{\beta V + n_{.|t}} \]

2. Quickly find \( t \)'s such that \( n_{w|t} \) is large for \( w \)

\[ z = s + r + q \]

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

store \( n_{w|t} \) for each word, topic pair

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

1. **Precompute**, for each \( t \),
   \[ \frac{\alpha_t + n_{t|d}}{\beta V + n_{.|t}} \]

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

- map \( w \) to 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...

**store** \( n_{w|t} \) for each word, topic pair ...???
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.
Outline

• LDA/Gibbs algorithm details
• How to speed it up by parallelizing
• How to speed it up by faster sampling
  – Why sampling is key
  – Some sampling ideas for LDA
    • The Mimno/McCallum decomposition (SparseLDA)
    • **Alias tables** (Walker 1977; Li, Ahmed, Ravi, Smola KDD 2014)
Alias tables

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 $r \sim \text{uniform}$ and use a binary tree

http://www.keithschwarz.com/darts-dice-coins/
Another idea…

Simulate the dart with two drawn values:

- \( rx \rightarrow \text{int}(u1*K) \)
- \( ry \rightarrow u1*p_{\text{max}} \)

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 a bit.

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/
Reducing the Sampling Complexity of Topic Models

Key ideas

• use variant of Mimno/McCallum decomposition

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

• Use alias tables to sample from the dense parts

• Since the alias table gradually goes stale, use Metropolis-Hastings sampling instead of Gibbs
Reducing the Sampling Complexity of Topic Models

StationaryMetropolisHastings\((p, q, n)\)

if no initial state exists then \(i \sim q(i)\)
for \(l = 1\) to \(n\) do
  Draw \(j \sim q(j)\)
  if \(\text{RandUnif}(1) < \min \left(1, \frac{p(j)q(i)}{p(i)q(j)}\right)\) then
    \(i \leftarrow j\)
  end if
end for
else the dart missed

- \(q\) is stale, easy-to-draw from distribution
- \(p\) is updated distribution
- computing ratios \(p(i)/q(i)\) is cheap
- usually the ratio is close to one
Reducing the Sampling Complexity of Topic Models

Table 1: Datasets and their statistics. V: vocabulary size; L: total number of training tokens, D: number of training documents; T: number of test documents. L/V is the average number occurrences of a word. L/D is the average document length.