Fast search for
Dirichlet process mixture models

Hal Daumé III
School of Computing
University of Utah
me@hal3.name
Dirichlet Process Mixture Models

- Non-parametric Bayesian density estimation
- Frequently used to solve clustering problem: choose “K”
- Applications:
  - vision
  - data mining
  - computational biology

Personal Observation:
Samplers slow on huge data sets (10k+ elements)
Very sensitive to initialization
Chinese Restaurant Process

- Customers enter a restaurant sequentially
- The $M$th customer chooses a table by:
  - Sit at table with $N$ customers with probability $N/(\alpha+M-1)$
  - Sit at unoccupied table with probability $\alpha/(\alpha+M-1)$
Dirichlet Process Mixture Models

- Data point = customer
- Cluster = table
- Each table gets a parameter
- Data points are generated according to a likelihood $F$

$$p(X | c) = \int d \theta_1:K \left[ \prod_k G_0(\theta_k) \right] \left[ \prod_n F(x_n | \theta_{c_n}) \right]$$

$c_n = \text{table of } n\text{th customer}$
Inference Summary

- Run MCMC sampler for a bunch of iterations
  - Use different initialization

- From set of samples, choose one with highest posterior probability

If all we want is the highest probability assignment, why not just try to find it directly?

*(If you really want to be Bayesian, use this assignment to initialize sampling)*
## Ordered Search

| **Input:** | data, beam size, scoring function |
| **Output:** | clustering |

- Initialize $Q$, a max-queue of partial clusterings
- While $Q$ is not empty
  - Remove a partial cluster $c$ from $Q$
  - If $c$ covers all the data, return it
  - Try extending $c$ by a single data point
  - Put all $K+1$ options into $Q$ with scores
  - If $|Q| > \text{beam size}$, drop elements

### Optimal, if:

- Beam size $= \infty$  
- Scoring function *overestimates* true best probability
Ordered Search in Pictures

Slide 7

Search for DPs
Trivial Scoring Function

- Only account for already-clustered data:

\[ g^{Triv} (c, x) = p_{\text{max}} (c) \prod_{k \in c} H \left( \{ x_{c=k} \} \right) \]

- \( p_{\text{max}} (c) \) can be computed exactly

\[ H (X) = \int d\theta G_0 (\theta) \prod_{x \in X} F (x | \theta) \]
Tighter Scoring Function

- Use trivial score for already-clustered data
- Approximate optimal score for future data:

  - For each data point, put in existing or new cluster
  - Then, conditioned on that choice, cluster remaining
    - Assume each remaining point is optimally placed
An Inadmissible Scoring Function

➢ Just use marginals for unclustered points:

\[ g^{\text{Inad}}(c, x) = g^{\text{Triv}}(c, x) \prod_{n=|c|+1}^N H(x_n) \]

➢ Inadmissible because \( H \) is not monotonic in conditioning (even for exponential family)
Artificial Data: Likelihoods Ratio

All that should be optimal, are.
Sampling is surprisingly unoptimal.
Inadmissible turns out to be optimal.
Artificial Data: Speed (Seconds)

Admissible is slow
Sampling is slower
Inadmissible is very fast
Artificial Data: # of Enqueued Points

Admissible functions enqueue a lot
Inadmissible enqueues almost nothing that is not required!
Real Data: MNIST

- Handwritten numbers 0-9, 28x28 pixels
- Preprocess with PCA to 50 dimensions
- Run on: 3000, 12,000 and 60,000 images
- Use inadmissible heuristic with (large) 100 beam

<table>
<thead>
<tr>
<th></th>
<th>3k</th>
<th>12k</th>
<th>60k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>11s</td>
<td>105s</td>
<td>15m</td>
</tr>
<tr>
<td></td>
<td>2.04e5</td>
<td>8.02e5</td>
<td>3.96e6</td>
</tr>
<tr>
<td>Gibbs</td>
<td>40s/i</td>
<td>18m/i</td>
<td>7h/i</td>
</tr>
<tr>
<td></td>
<td>2.09e5</td>
<td>8.34e5</td>
<td>4.2e6</td>
</tr>
<tr>
<td>S-M</td>
<td>85s/i</td>
<td>35m/i</td>
<td>12h/i</td>
</tr>
<tr>
<td></td>
<td>2.05e5</td>
<td>8.15e5</td>
<td>4.1e6</td>
</tr>
</tbody>
</table>
Real Data: NIPS Papers

- NIPS 1-12
- 1740 documents, vocabulary of 13k words
  - Drop top 10, retain remaining top 1k
- Conjugate Dirichlet/Multinomial DP
- Order examples by increasing marginal likelihood

```
<table>
<thead>
<tr>
<th>Method</th>
<th>Count</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>2.441e6</td>
<td>32s</td>
</tr>
<tr>
<td></td>
<td>2.474e6 (reverse order)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.449e6 (random order)</td>
<td></td>
</tr>
<tr>
<td>Gibbs</td>
<td>3.2e6</td>
<td>1h</td>
</tr>
<tr>
<td>S-M</td>
<td>3.0e6</td>
<td>1.5h</td>
</tr>
</tbody>
</table>
```
Discussion

Sampling often fails to find MAP

Search can do much better

Limited to conjugate distributions

Cannot re-estimate hyperparameters

Can cluster 270 images / second in matlab

Further acceleration possible with clever data structures

Thanks! Questions?

code at http://hal3.name/DPsearch
Inference I – Gibbs Sampling

Collapsed Gibbs sampler:

- Initialize clusters
- For a number of iterations:
  - Assign each data point $x_n$ to an existing cluster $c_k$ with probability:
    
    $$
    \frac{N_k}{\alpha + N - 1} \int d\theta G_0(\theta) F(x_n|\theta) \prod_{m \in c_k} F(x_m|\theta)
    $$

  - or to a new cluster with probability
    
    $$
    \frac{\alpha}{\alpha + N - 1} \int d\theta G_0(\theta) F(x_n|\theta)
    $$

$H$ is the posterior probability of $x$, conditioned on the set of $x$ that fall into the proposed cluster.
Inference II – Metropolis-Hastings

Collapsed Split-Merge sampler:

➢ Initialize clusters
➢ For a number of iterations:
  ➢ Choose two data points $x_n$ and $x_m$ at random
    ➢ If $c_n = c_m$, split this cluster with a Gibbs pass
    ➢ otherwise, merge the two clusters
  ➢ Then perform a collapsed Gibbs pass
Versus Variational