Fast search for Dirichlet process mixture models

Hal Daumé III

School of Computing
University of Utah

me@hal3.name

THE UNIVERSITY OF UTAH

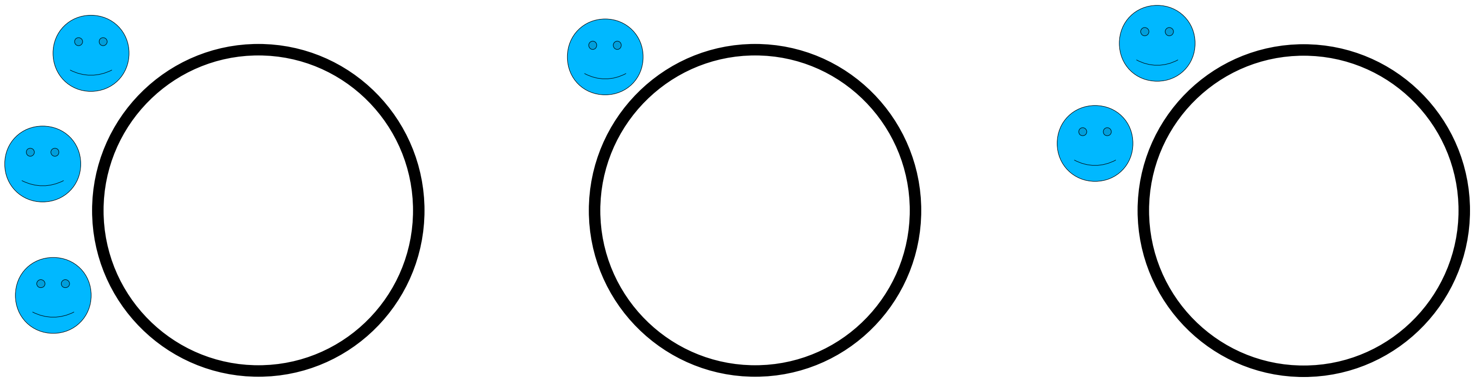
Dirichlet Process Mixture Models

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

Personal Observation:

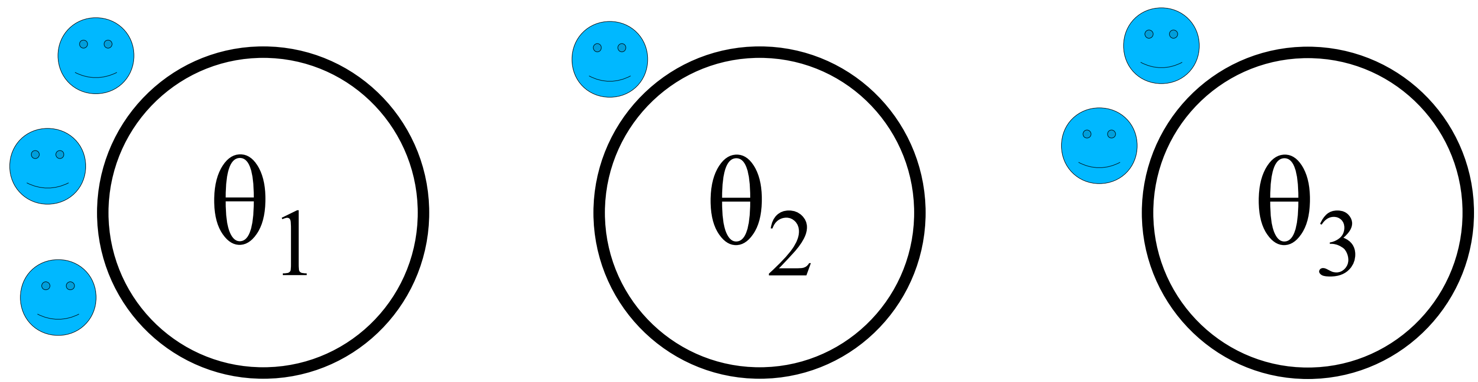
Samplers slow on huge data sets (10k+ elements)
Very sensitive to initialization

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/(α+M-1)
  - Sit at unoccupied table with probability α/(α+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

$$p(X \mid c) = \int d\theta_{1:K} \left[\prod_k G_0(\theta_k) \right] \left[\prod_n F(x_n \mid \theta_{c_n}) \right]$$

c_n = table of nth 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

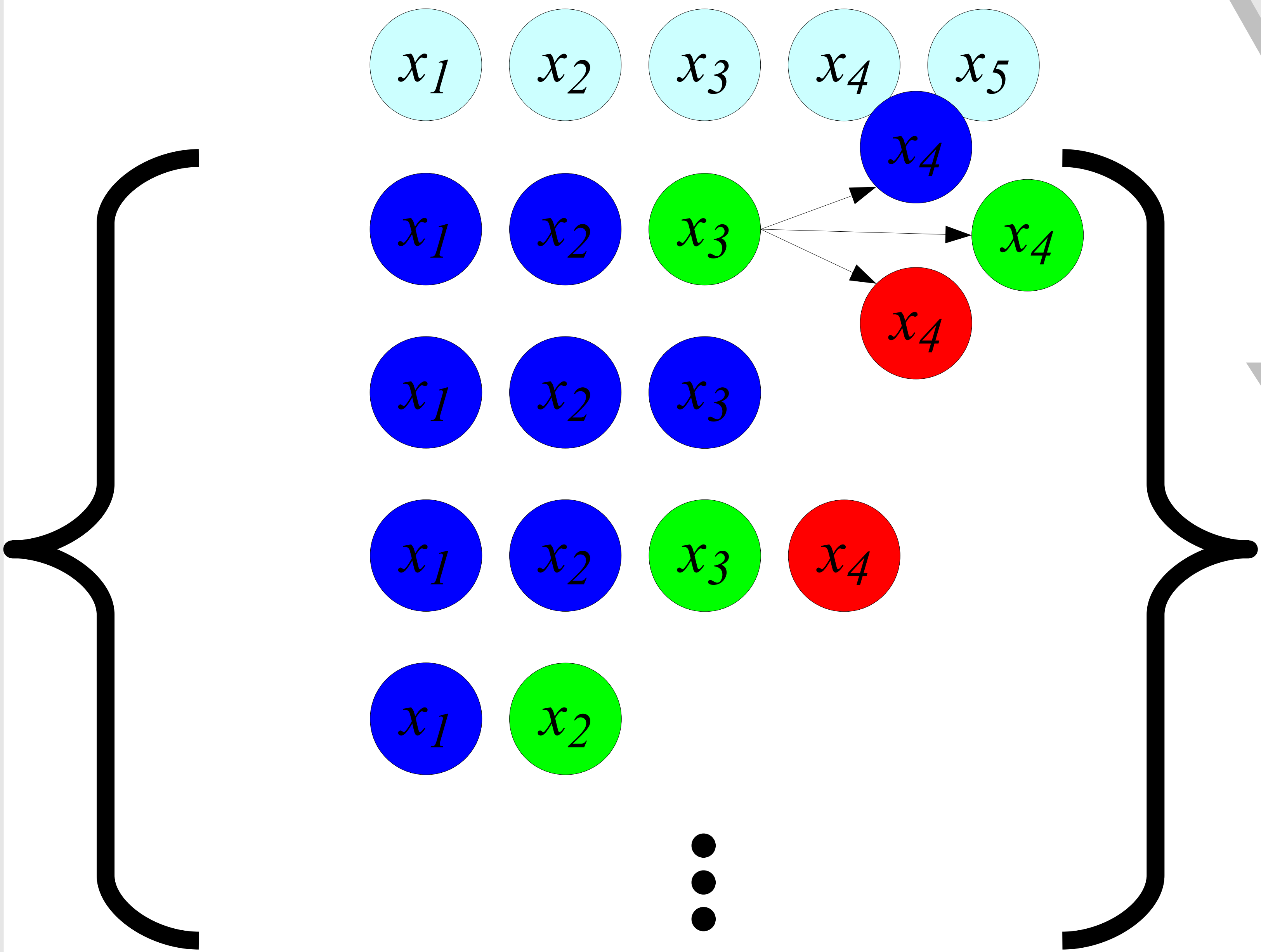
initialize **Q**, a max-queue of partial clusterings
while **Q** is not empty
    remove a partial cluster **c** from **Q**
    if **c** covers all the data, return it
    try extending **c** by a single data point
    put all K+1 options into **Q** with scores
    if |Q| > beam size, drop elements

Optimal, if:

beam size = infinity

scoring function *overestimates* true best probability

Ordered Search in Pictures



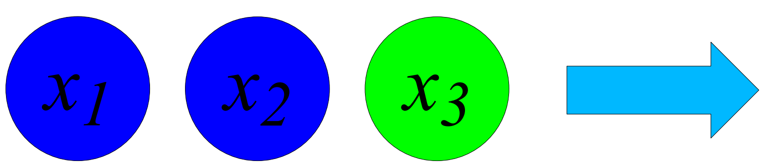
Trivial Scoring Function

- Only account for already-clustered data:

$$g^{Triv}(c, x) = p_{max}(c) \prod_{k \in c} H(\{x_{c=k}\})$$

$$H(X) = \int d\theta \, G_0(\theta) \prod_{x \in X} F(x \mid \theta)$$

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



$$p_{max}(c) H(x_1, x_2) H(x_3)$$

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

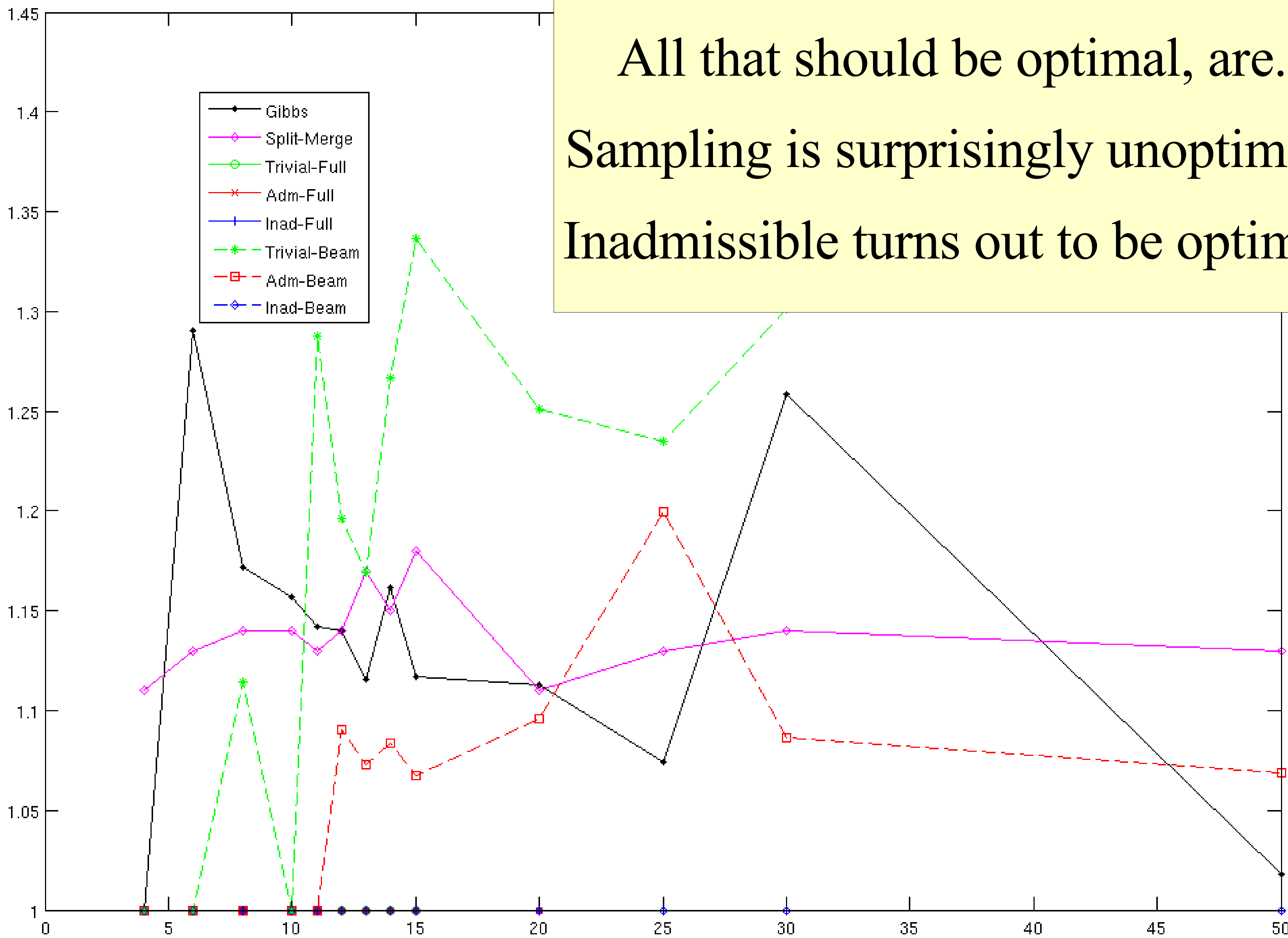
An Inadmissible Scoring Function

- Just use marginals for unclustered points:

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

- Inadmissible because H is not monotonic in conditioning (even for exponential family)

Artificial Data: Likelihoods Ratio

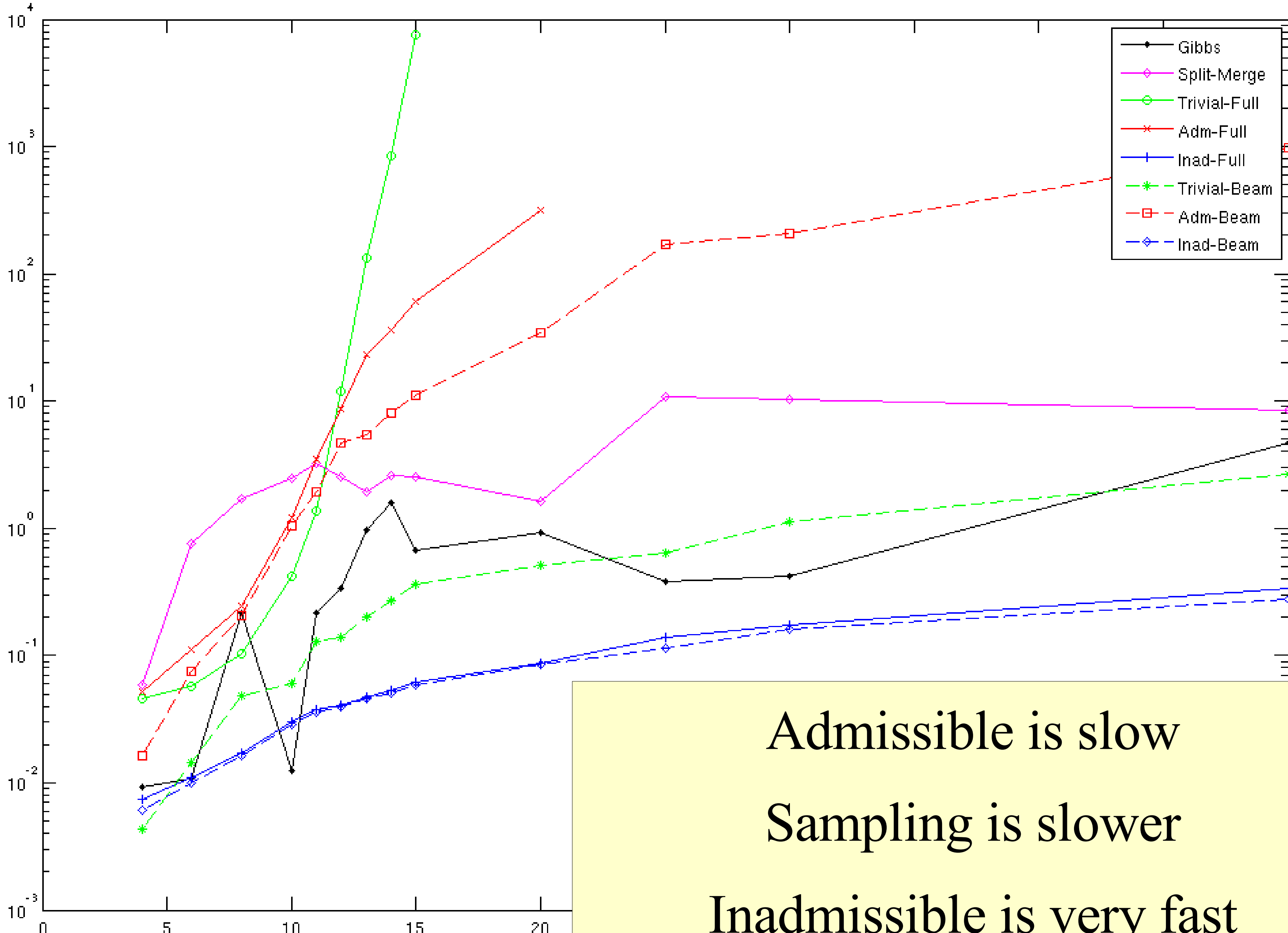


All that should be optimal, are.

Sampling is surprisingly unoptimal.

Inadmissible turns out to be optimal

Artificial Data: Speed (Seconds)



Admissible is slow

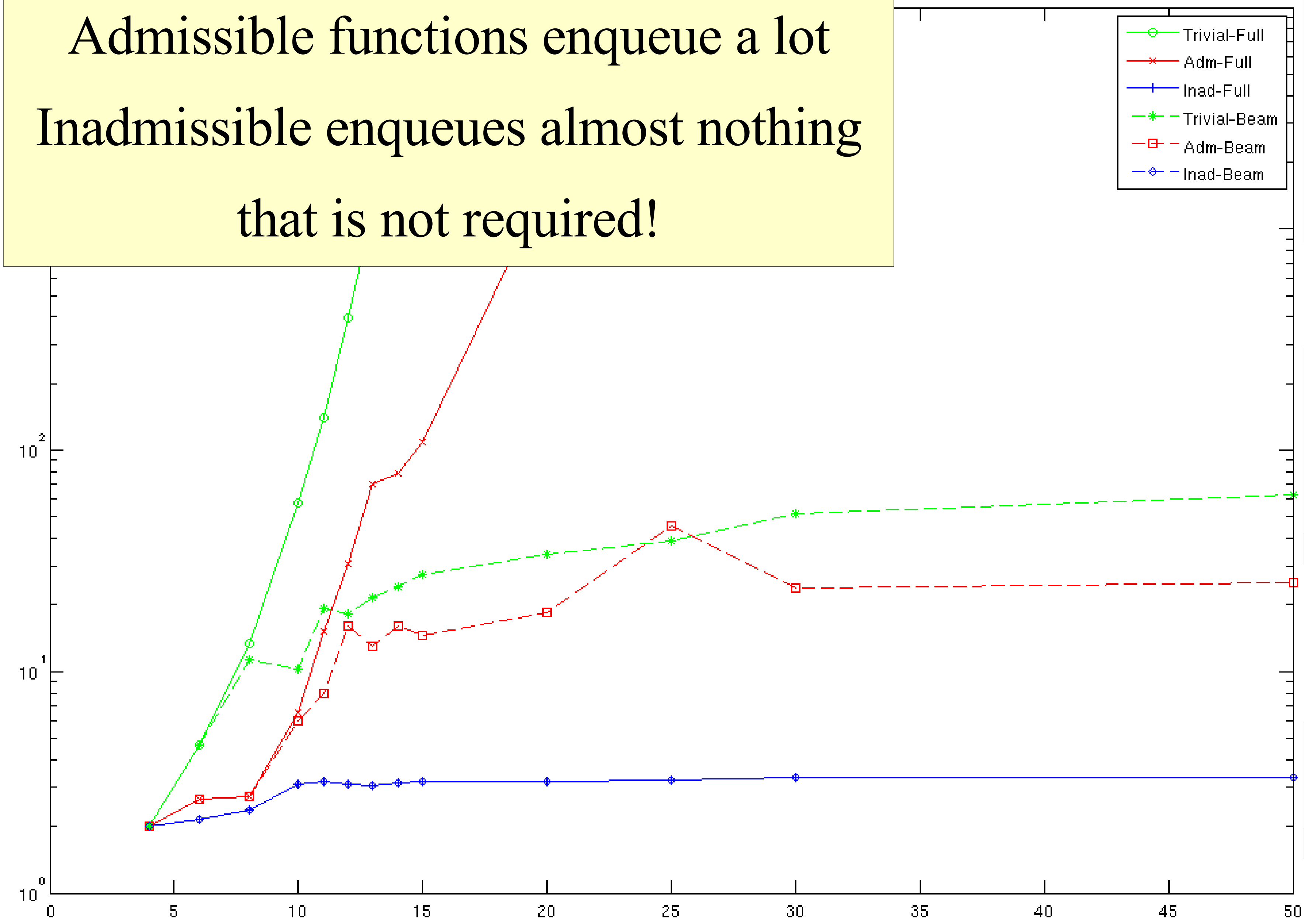
Sampling is slower

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

	3k	12k	60k
Search	11s *2.04e5*	105s *8.02e5*	15m *3.96e6*
Gibbs	40s/i *2.09e5*	18m/i *8.34e5*	7h/i *4.2e6*
S-M	85s/i *2.05e5*	35m/i *8.15e5*	12h/i *4.1e6*

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

Search	*2.441e6* (32s) *2.474e6 (reverse order)* *2.449e6 (random order)*
Gibbs	*3.2e6* (1h)
S-M	*3.0e6* (1.5h)

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:

$$\frac{N_k}{\alpha + N - 1} \int d\theta \, G_0(\theta) F(x_n|\theta) \prod_{m \in c_k} F(x_m|\theta)$$

$$= H\left(x_n \mid \{x_{c=k}\}\right)$$

H is the posterior probability of x, conditioned on the set of x that fall into the proposed cluster

or to a new cluster with probability

$$\frac{\alpha}{\alpha + N - 1} \int d\theta \, G_0(\theta) F(x_n|\theta)$$

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