Graphical Models for the Internet

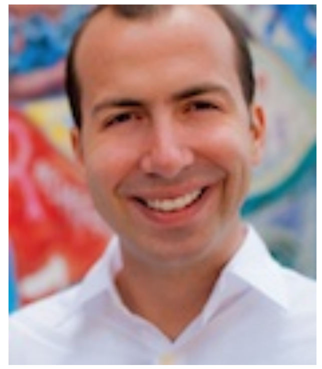
Amr Ahmed & Alexander Smola

Yahoo! Research & Australian National University, Santa Clara, CA

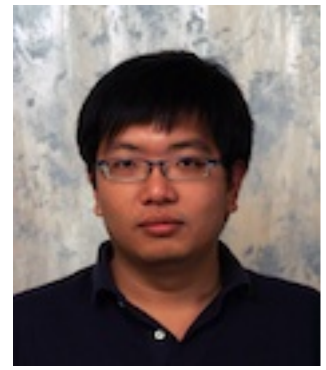
amahmed@cs.cmu.edu, alex@smola.org

Thanks

Mohamed Aly



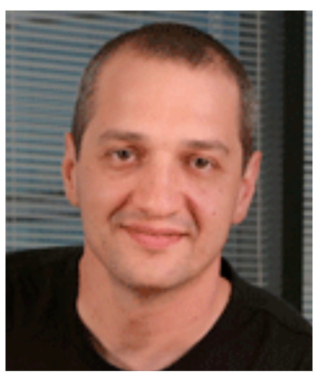
Joey Gonzalez



Yucheng Low

Qirong Ho

Shravan Narayanamurthy



Vanja Josifovski

Choon Hui Teo

Eric Xing

James Petterson

Sergiy Matyusevich

Jake Eisenstein

Shuang Hong Yang

Vishy Vishwanathan

Markus Weimer

Alexandros Karatzoglou

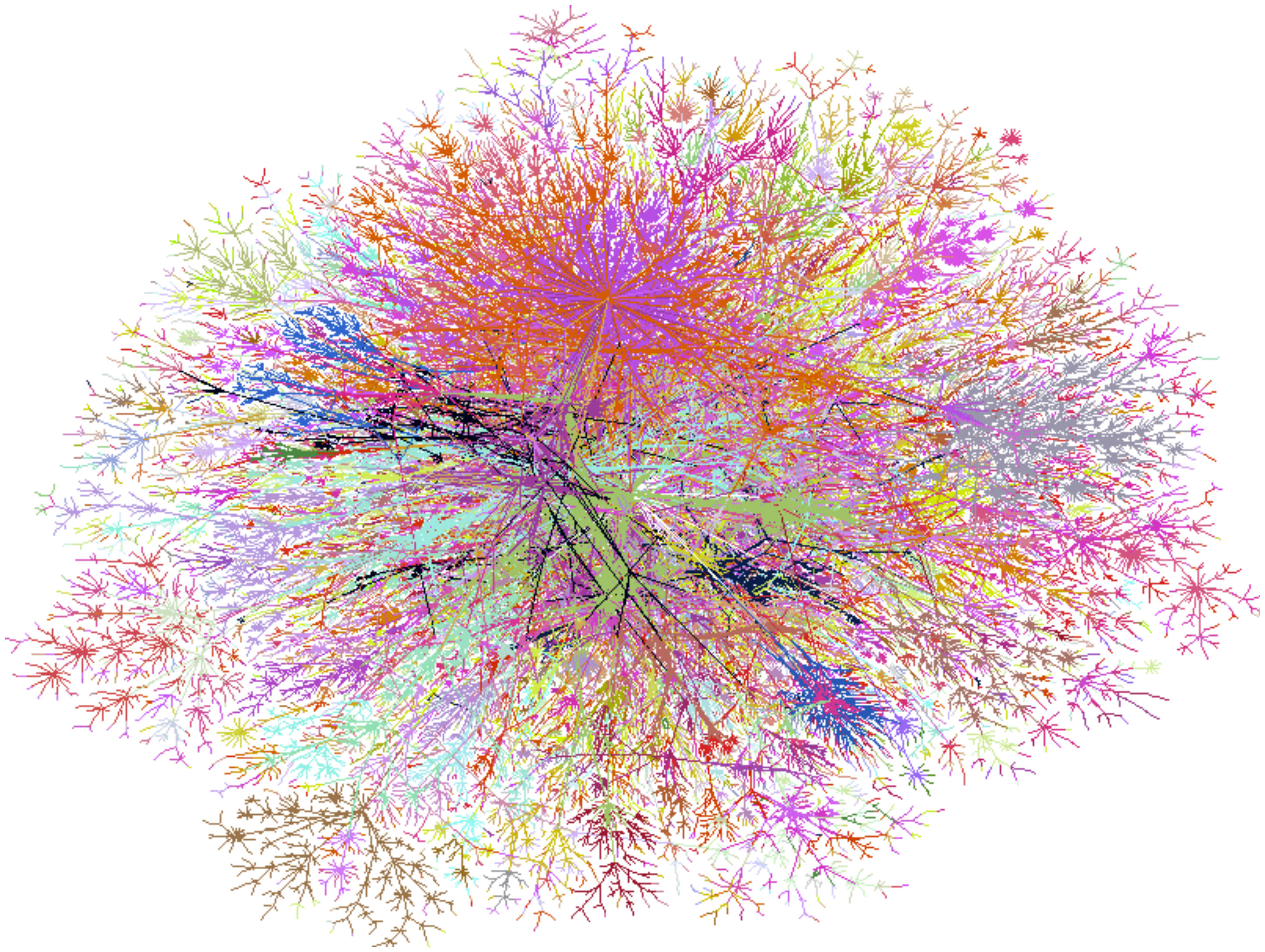
1. Data on the Internet

Size calibration

- **Tiny** (2 cores)
  (512MB, 50MFlops, 1000 samples)
- **Small** (4 cores)
  (4GB, 10GFlops, 100k samples)
- **Medium** (16 cores)
  (32GB, 100GFlops, 1M samples)
- **Large** (256 cores)
  (512GB, 1TFlops, 100M samples)
- **Massive**
  ... need to work hard get it to work

Data

- Webpages (content, graph)
- Clicks (ad, page, social)
- Users (OpenID, FB Connect)
- e-mails (Hotmail, Y!Mail, Gmail)
- Photos, Movies (Flickr, YouTube, Vimeo ...)
- Cookies / tracking info (see Ghostery)
- Installed apps (Android market etc.)
- Location (Latitude, Loopt, Foursquared)
- User generated content (Wikipedia & co)
- Ads (display, text, DoubleClick, Yahoo)
- Comments (Disqus, Facebook)
- Reviews (Yelp, Y!Local)
- Third party features (e.g. Experian)
- Social connections (LinkedIn, Facebook)
- Purchase decisions (Netflix, Amazon)
- Instant Messages (YIM, Skype, Gtalk)
- Search terms (Google, Bing)
- Timestamp (everything)
- News articles (BBC, NYTimes, Y!News)
- Blog posts (Tumblr, Wordpress)
- Microblogs (Twitter, Jaiku, Meme)



>10B useful webpages

The Web for $100k/month

- Webpages (content, graph)
- Clicks (ad, page, social)
- Users (OpenID, FB Connect)
- e-mails (Hotmail, Y!Mail, Gmail)
- Photos, Movies (Flickr, YouTube, Vimeo ...)
- Cookies / tracking info (see Ghostery)
- Installed apps (Android market etc.)
- Location (Latitude, Loopt, Foursquared)
- User generated content (Wikipedia & co)
- Ads (display, text, DoubleClick, Yahoo)
- Comments (Disqus, Facebook)
- Reviews (Yelp, Y!Local)
- Third party features (e.g. Experian)
- Social connections (LinkedIn, Facebook)
- Purchase decisions (Netflix, Amazon)
- Instant Messages (YIM, Skype, Gtalk)
- Search terms (Google, Bing)
- Timestamp (everything)
- News articles (BBC, NYTimes, Y!News)
- Blog posts (Tumblr, Wordpress)
- Microblogs (Twitter, Jaiku, Meme)

- **10 billion pages**
  (this is a small subset, maybe 10%)
  10k/page = 100TB
  ($10k for disks or EBS 1 month)
- **1000 machines**
  10ms/page = 1 day
  afford 1-10 MIP/page
  ($20k on EC2 for 0.68$/h)
- **10 Gbit link**
  ($10k/month via ISP or EC2)
  - 1 day for raw data
  - 300ms/page roundtrip
  - 1000 servers for 1 month
    ($70k on EC2 for 0.085$/h)

Data - Identity & Graph

- Webpages (content, graph)
- Clicks (ad, page, social)
- Users (OpenID, FB Connect)
- e-mails (Hotmail, Y!Mail, Gmail)
- Photos, Movies (Flickr, YouTube, Vimeo ...)
- Cookies / tracking info (see Ghostery)
- Installed apps (Android market etc.)
- Location (Latitude, Loopt, Foursquared)
- User generated content (Wikipedia & co)
- Ads (display, text, DoubleClick, Yahoo)
- Comments (Disqus, Facebook)
- Reviews (Yelp, Y!Local)
- Third party features (e.g. Experian)
- Social connections (LinkedIn, Facebook)
- Purchase decisions (Netflix, Amazon)
- Instant Messages (YIM, Skype, Gtalk)
- Search terms (Google, Bing)
- Timestamp (everything)
- News articles (BBC, NYTimes, Y!News)
- Blog posts (Tumblr, Wordpress)
- Microblogs (Twitter, Jaiku, Meme)

100M-1B vertices

Data - User generated content

- Webpages (content, graph)
- Clicks (ad, page, social)
- Users (OpenID, FB Connect)
- e-mails (Hotmail, Y!Mail, Gmail)
- Photos, Movies (Flickr, YouTube, Vimeo ...)
- Cookies / tracking info (see Ghostery)
- Installed apps (Android market etc.)
- Location (Latitude, Loopt, Foursquared)
- User generated content (Wikipedia & co)
- Ads (display, text, DoubleClick, Yahoo)
- Comments (Disqus, Facebook)
- Reviews (Yelp, Y!Local)
- Third party features (e.g. Experian)
- Social connections (LinkedIn, Facebook)
- Purchase decisions (Netflix, Amazon)
- Instant Messages (YIM, Skype, Gtalk)
- Search terms (Google, Bing)
- Timestamp (everything)
- News articles (BBC, NYTimes, Y!News)
- Blog posts (Tumblr, Wordpress)
- Microblogs (Twitter, Jaiku, Meme)

>1B images, 40h video/minute

Data - User generated content

- Webpages (content, graph)
- Clicks (ad, page, social)
- Users (OpenID, FB Connect)
- e-mails (Hotmail, Y!Mail, Gmail)
- Photos, Movies (Flickr, YouTube, Vimeo ...)
- Cookies / tracking info (see Ghostery)
- Installed apps (Android market etc.)
- Location (Latitude, Loopt, Foursquared)
- User generated content (Wikipedia & co)
- Ads (display, text, DoubleClick, Yahoo)
- Comments (Disqus, Facebook)
- Reviews (Yelp, Y!Local)
- Third party features (e.g. Experian)
- Social connections (LinkedIn, Facebook)
- Purchase decisions (Netflix, Amazon)
- Instant Messages (YIM, Skype, Gtalk)
- Search terms (Google, Bing)
- Timestamp (everything)
- News articles (BBC, NYTimes, Y!News)
- Blog posts (Tumblr, Wordpress)
- Microblogs (Twitter, Jaiku, Meme)

crawl it

>1B images, 40h video/minute

Data - Messages

- Webpages (content, graph)
- Clicks (ad, page, social)
- Users (OpenID, FB Connect)
- e-mails (Hotmail, Y!Mail, Gmail)
- Photos, Movies (Flickr, YouTube, Vimeo ...)
- Cookies / tracking info (see Ghostery)
- Installed apps (Android market etc.)
- Location (Latitude, Loopt, Foursquared)
- User generated content (Wikipedia & co)
- Ads (display, text, DoubleClick, Yahoo)
- Comments (Disqus, Facebook)
- Reviews (Yelp, Y!Local)
- Third party features (e.g. Experian)
- Social connections (LinkedIn, Facebook)
- Purchase decisions (Netflix, Amazon)
- Instant Messages (YIM, Skype, Gtalk)
- Search terms (Google, Bing)
- Timestamp (everything)
- News articles (BBC, NYTimes, Y!News)
- Blog posts (Tumblr, Wordpress)
- Microblogs (Twitter, Jaiku, Meme)

>1B texts

Data - Messages

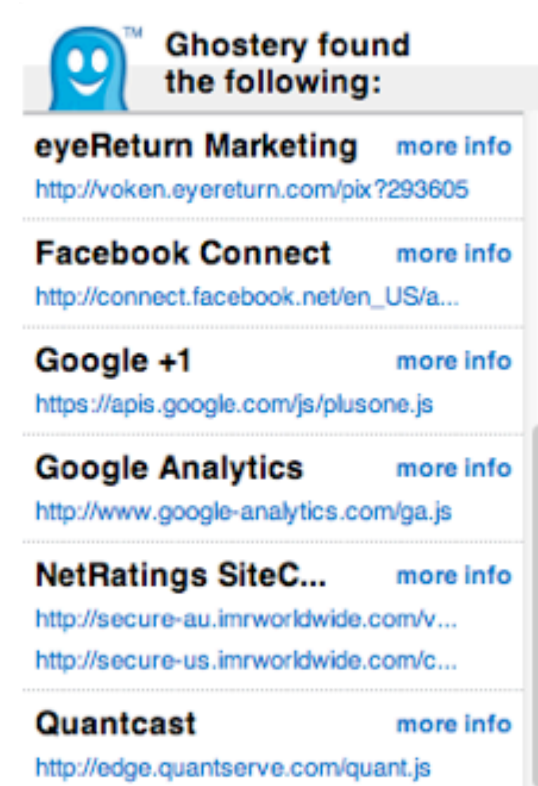
- Webpages (content, graph)
- Clicks (ad, page, social)
- Users (OpenID, FB Connect)
- e-mails (Hotmail, Y!Mail, Gmail)
- Photos, Movies (Flickr, YouTube, Vimeo ...)
- Cookies / tracking info (see Ghostery)
- Installed apps (Android market etc.)
- Location (Latitude, Loopt, Foursquared)
- User generated content (Wikipedia & co)
- Ads (display, text, DoubleClick, Yahoo)
- Comments (Disqus, Facebook)
- Reviews (Yelp, Y!Local)
- Third party features (e.g. Experian)
- Social connections (LinkedIn, Facebook)
- Purchase decisions (Netflix, Amazon)
- Instant Messages (YIM, Skype, Gtalk)
- Search terms (Google, Bing)
- Timestamp (everything)
- News articles (BBC, NYTimes, Y!News)
- Blog posts (Tumblr, Wordpress)
- Microblogs (Twitter, Jaiku, Meme)

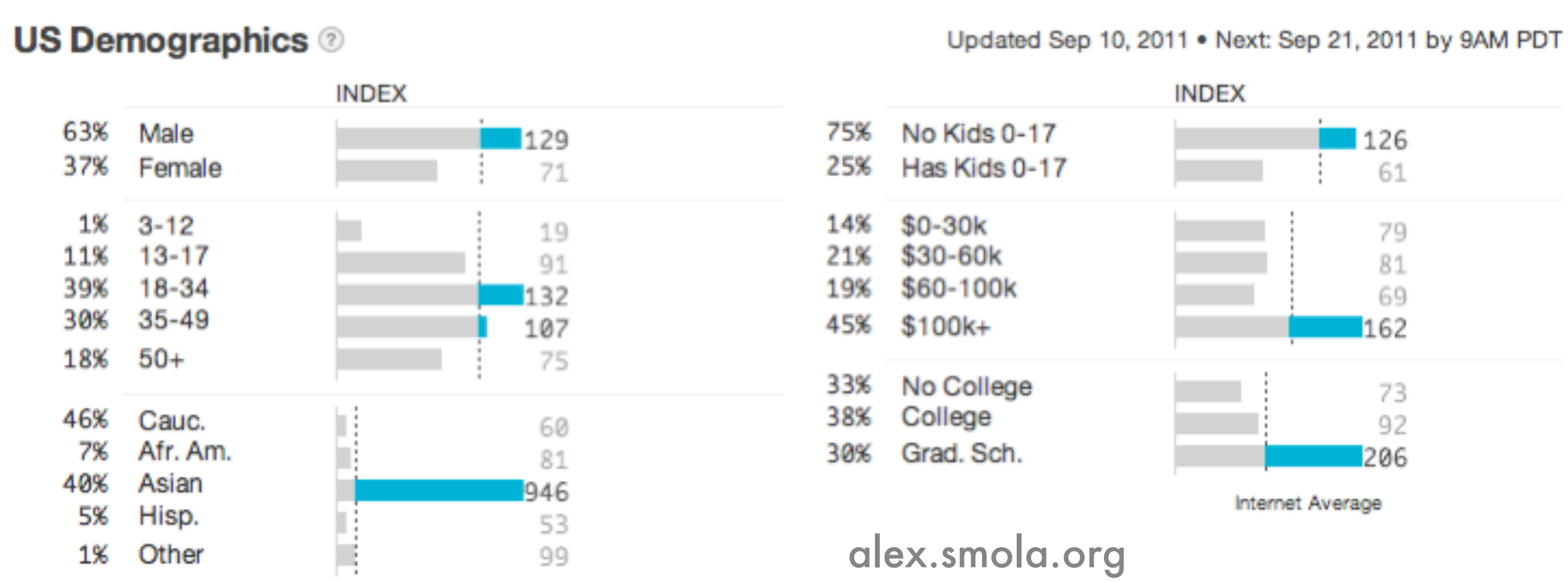
>1B texts

impossible without NDA

Data - User Tracking

- Webpages (content, graph)
- Clicks (ad, page, social)
- Users (OpenID, FB Connect)
- e-mails (Hotmail, Y!Mail, Gmail)
- Photos, Movies (Flickr, YouTube, Vimeo ...)
- Cookies / tracking info (see Ghostery)
- Installed apps (Android market etc.)
- Location (Latitude, Loopt, Foursquared)
- User generated content (Wikipedia & co)
- Ads (display, text, DoubleClick, Yahoo)
- Comments (Disqus, Facebook)
- Reviews (Yelp, Y!Local)
- Third party features (e.g. Experian)
- Social connections (LinkedIn, Facebook)
- Purchase decisions (Netflix, Amazon)
- Instant Messages (YIM, Skype, Gtalk)
- Search terms (Google, Bing)
- Timestamp (everything)
- News articles (BBC, NYTimes, Y!News)
- Blog posts (Tumblr, Wordpress)
- Microblogs (Twitter, Jaiku, Meme)





alex.smola.org

>1B 'identities'

Data - User Tracking

- Webpages (content, graph)
- Clicks (ad, page, social)
- Users (OpenID, FB Connect)
- e-mails (Hotmail, Y!Mail, Gmail)
- Photos, Movies (Flickr, YouTube, Vimeo ...)
- Cookies / tracking info (see Ghostery)
- Installed apps (Android market etc.)
- Location (Latitude, Loopt, Foursquared)
- User generated content (Wikipedia & co)
- Ads (display, text, DoubleClick, Yahoo)
- Comments (Disqus, Facebook)
- Reviews (Yelp, Y!Local)
- Third party features (e.g. Experian)
- Social connections (LinkedIn, Facebook)
- Purchase decisions (Netflix, Amazon)
- Instant Messages (YIM, Skype, Gtalk)
- Search terms (Google, Bing)
- Timestamp (everything)
- News articles (BBC, NYTimes, Y!News)
- Blog posts (Tumblr, Wordpress)
- Microblogs (Twitter, Jaiku, Meme)

Privacy Information

Privacy Policy:

http://www.facebook.com/policy.php

Data Collected:

Anonymous (browser type, location, page views), Pseudonymous (IP address, "actions taken")

Data Sharing:

Data is shared with third parties.

Data Retention:

Data is deleted from backup storage after 90 days.

Privacy Information

Privacy Policy:

http://www.google.com/intl/en/priv...

Data Collected:

Anonymous (ad serving domains, browser type, demographics, language settings, page views, time/date), Pseudonymous (IP address)

Data Sharing:

Anonymous data is shared with third parties.

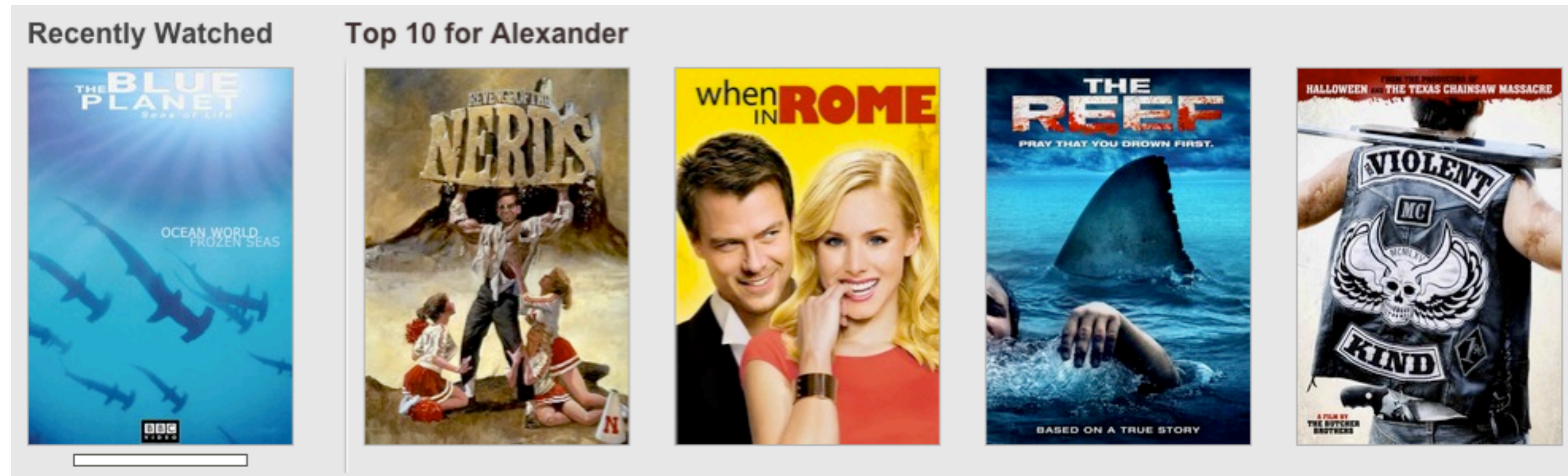
Data Retention:

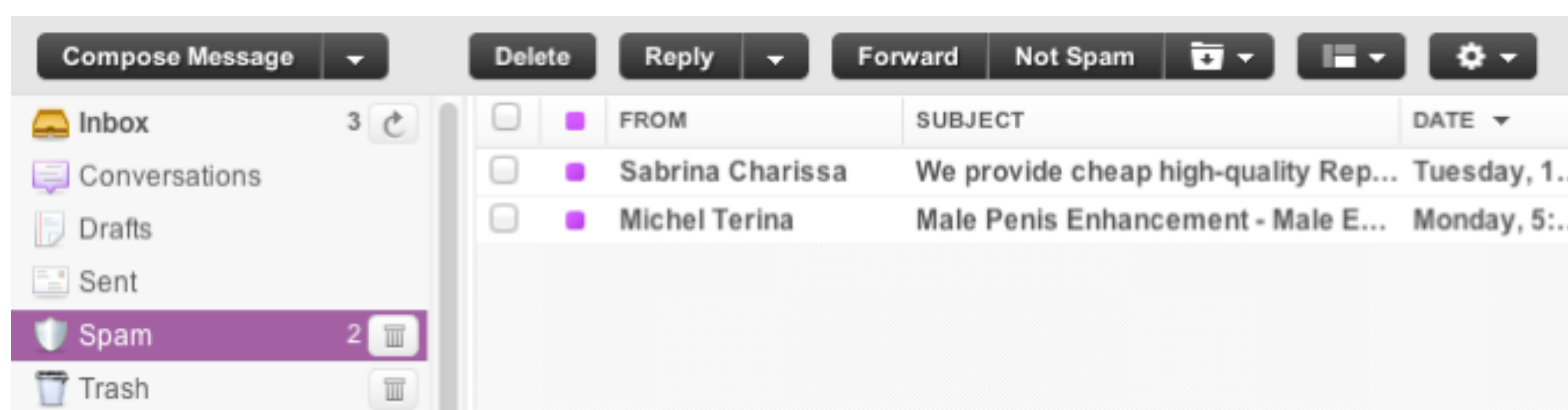
Undisclosed

Personalization

- 100-1000M users
  - Spam filtering
  - Personalized targeting & collaborative filtering
  - News recommendation
  - Advertising

- Large parameter space (25 parameters = 100GB)
- Distributed storage (need it on every server)
- Distributed optimization
- Model synchronization
- Time dependence
- Graph structure

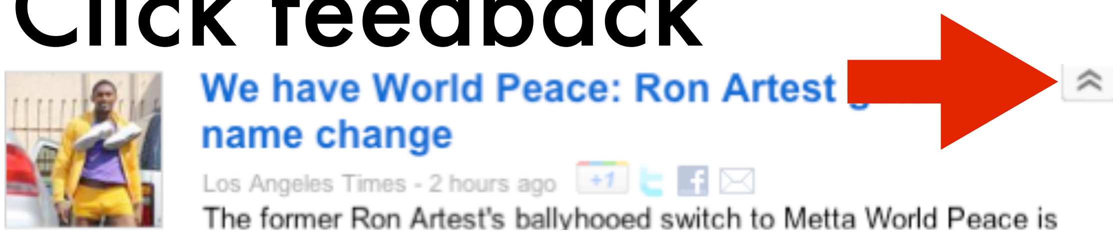




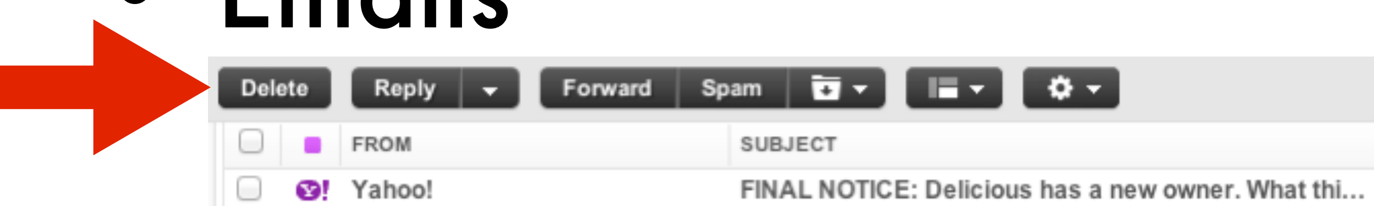
(implicit) Labels

- Ads

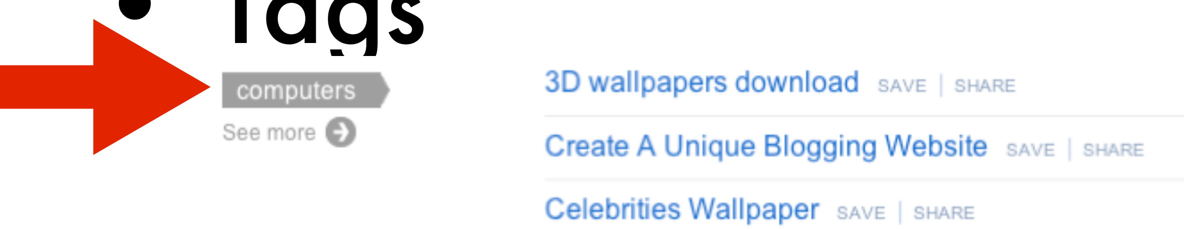
- Click feedback



- Emails



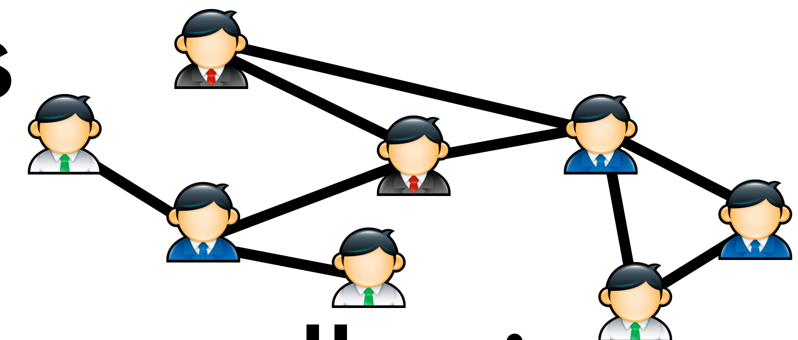
- Tags



- Editorial data is very expensive! Do not use!

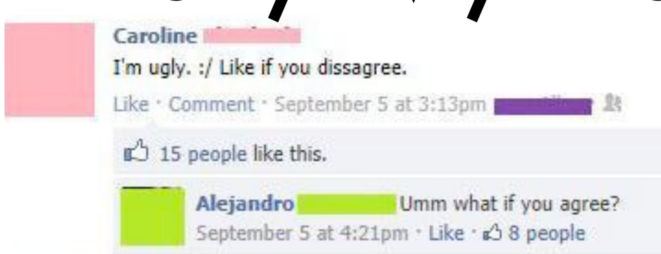
no labels (typical case)

- Graphs

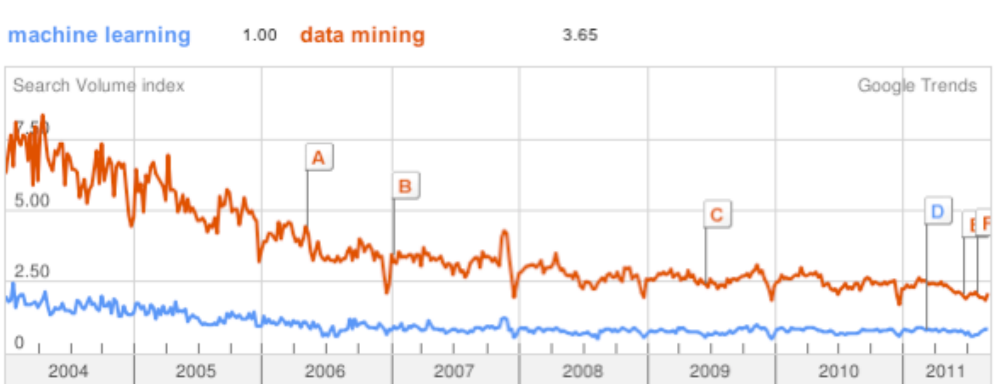


- Document collections

- Email/IM/Discussions



- Query stream



Challenges

- Scale
  - Billions of instances (documents, clicks, users, ads)
  - Rich data structure (ontology, categories, tags)
  - Model does not fit into single machine
- Modeling
  - Plenty of unlabeled data, temporal structure, side information
  - User-understandable structure
  - Solve problem. Don’t simply apply clustering/LDA/PCA/ICA
  - We only cover building blocks
- Inference
  - 10k-100k clusters/discrete objects, 1M-100M unique tokens
  - Communication

Roadmap

- Tools
  - Load distribution, balancing and synchronization
  - Clustering, Topic Models
- Models
  - Dynamic non-parametric models
  - Sequential latent variable models
- Inference Algorithms
  - Distributed batch
  - Sequential Monte Carlo
- Applications
  - User profiling
  - News content analysis & recommendation

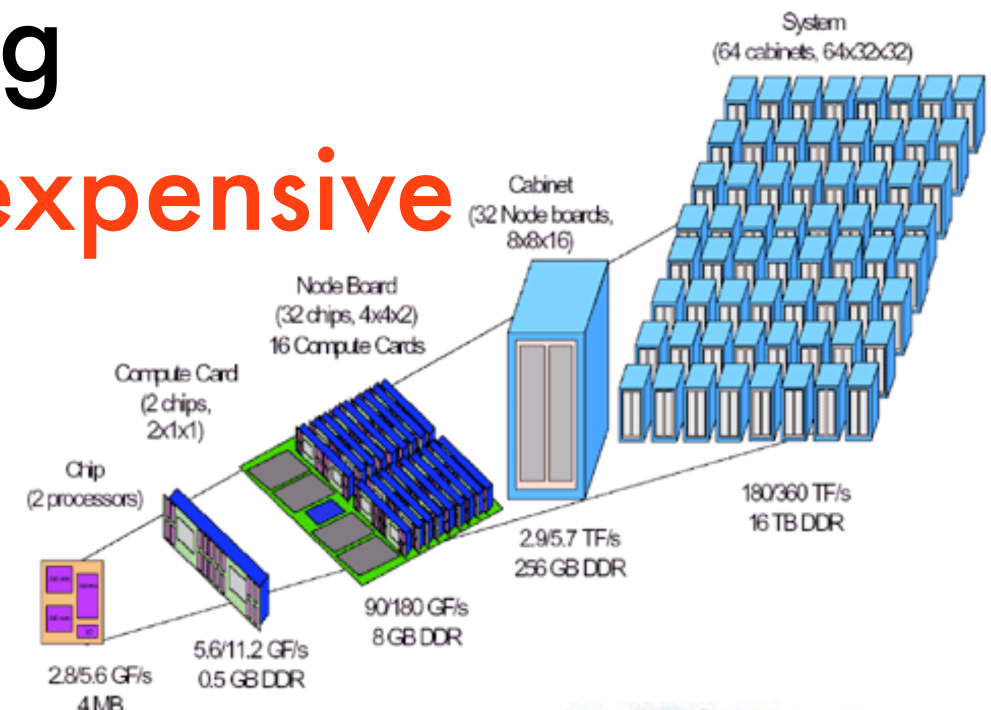
2. Basic Tools

Systems

Algorithms run on **MANY REAL and FAULTY boxes** not Turing machines. So we need to deal with it.

Commodity Hardware

- High Performance Computing
  Very reliable, custom built, expensive



- Consumer hardware
  Cheap, efficient, easy to replicate,
  Not very reliable, deal with it!

The Joys of Real Hardware

Typical first year for a new cluster:

~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)
~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)
~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)
~1 network rewiring (rolling ~5% of machines down over 2-day span)
~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)
~5 racks go wonky (40-80 machines see 50% packetloss)
~8 network maintenances (4 might cause ~30-minute random connectivity losses)
~12 router reloads (takes out DNS and external vips for a couple minutes)
~3 router failures (have to immediately pull traffic for an hour)
~dozens of minor 30-second blips for dns
~1000 individual machine failures
~thousands of hard drive failures

slow disks, bad memory, misconfigured machines, flaky machines, etc.

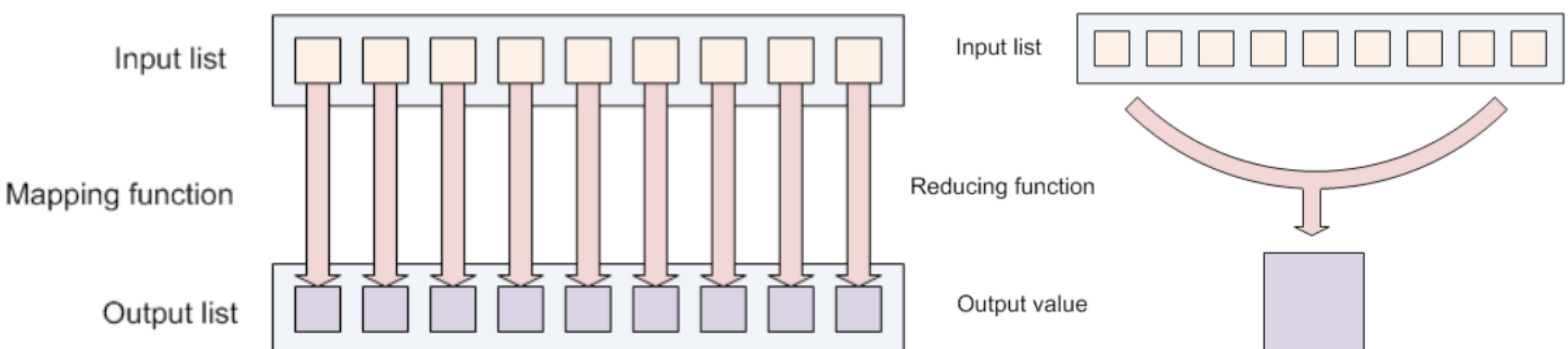
Google

Slide from talk of Jeff Dean

http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en//people/jeff/stanford-295-talk.pdf

Map Reduce

- 1000s of (faulty) machines
- Lots of jobs are mostly embarrassingly parallel (except for a sorting/transpose phase)
- Functional programming origins
  - Map(key,value)
    processes each (key,value) pair and outputs a new (key,value) pair
  - Reduce(key,value)
    reduces all instances with same key to aggregate



from Ramakrishnan, Sakrejda, Canon, DoE 2011

Map Reduce

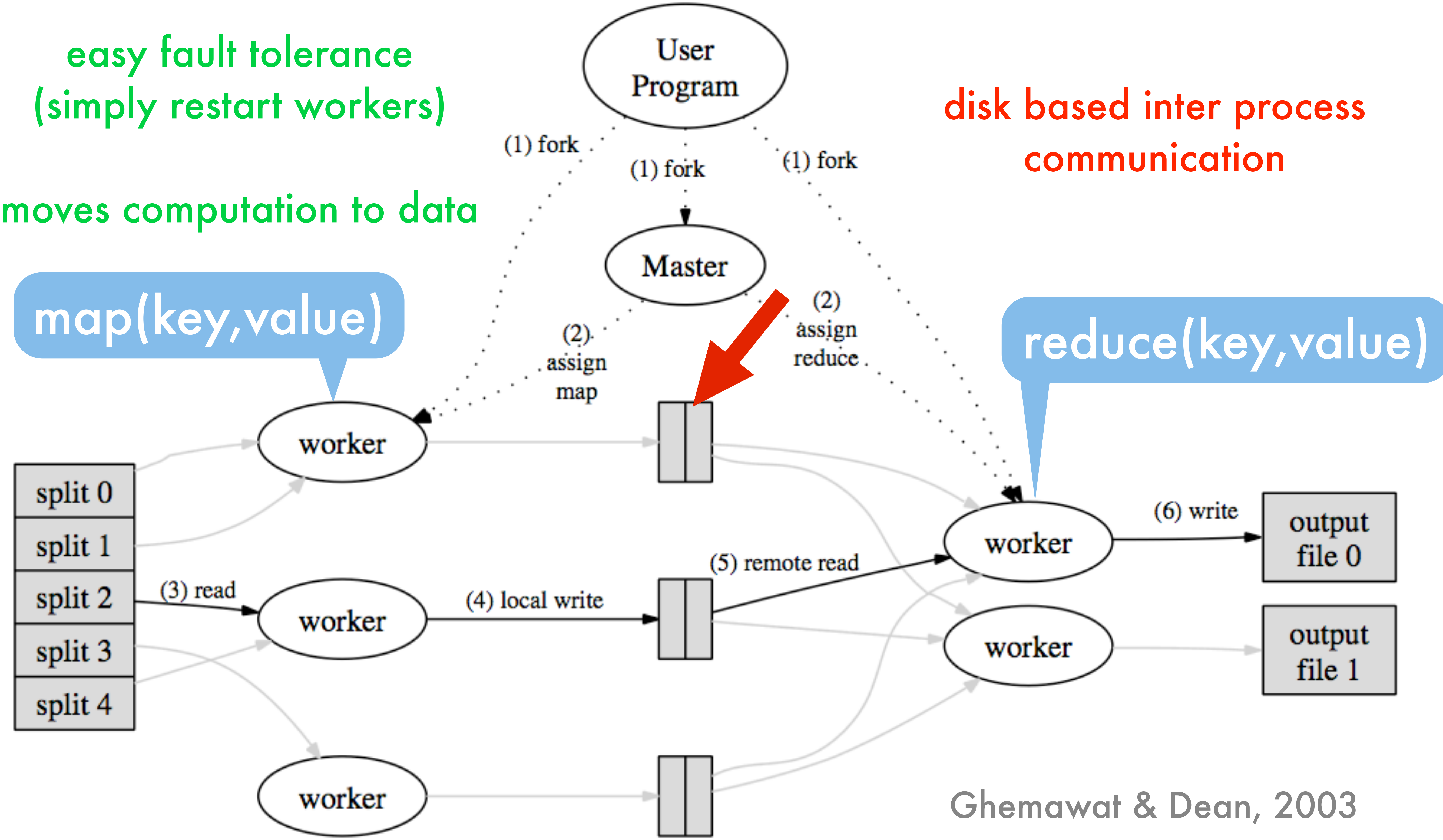
- 1000s of (faulty) machines
- Lots of jobs are mostly embarrassingly parallel
  (except for a sorting/transpose phase)
- Functional programming origins
  - Map(key,value)
    processes each (key,value) pair and outputs a new (key,value) pair
  - Reduce(key,value)
    reduces all instances with same key to aggregate
- Example - extremely naive wordcount
  - Map(docID, document)
    for each document emit many (wordID, count) pairs
  - Reduce(wordID, count)
    sum over all counts for given wordID and emit (wordID, aggregate)

Map Reduce

easy fault tolerance
(simply restart workers)

moves computation to data

disk based inter process communication



Ghemawat & Dean, 2003

Map Reduce

- Map must be stateless in blocks
- Reduce must be commutative in data
- Fault tolerance
  - Start jobs where the data is (nodes run the filesystem, too)
  - Restart machines if maps fail (have replicas)
  - Restart reducers based on intermediate data
- Good fit for many algorithms
- Good if only a small number of MapReduce iterations needed
- Need to request machines at each iteration (time consuming)
- State lost in between maps
- Communication only via file I/O
- Need to wait for last reducer

Map Reduce

- Map must be stateless in blocks
- Reduce must be commutative in data
- Fault tolerance
  - Start jobs where the data is (nodes run the filesystem, too)
  - Restart machines if maps fail (have replicas)
  - Restart reducers based on intermediate data
- Good fit for many algorithms
- Good if only a small number of MapReduce iterations needed
- Need to request machines at each iteration (time consuming)
- State lost in between maps
- Communication only via file I/O
- Need to wait for last reducer

unsuitable for algorithms with many iterations

Many alternatives

- Dryad/LINQ
  Microsoft - directed acyclic graphs
- S4
  Yahoo - streaming directed acyclic graphs
- Pregel
  Google - bulk synchronous processing
- YARN
  Use Hadoop scheduler directly
- Mesos, Hadoop workalikes & patches

Clustering

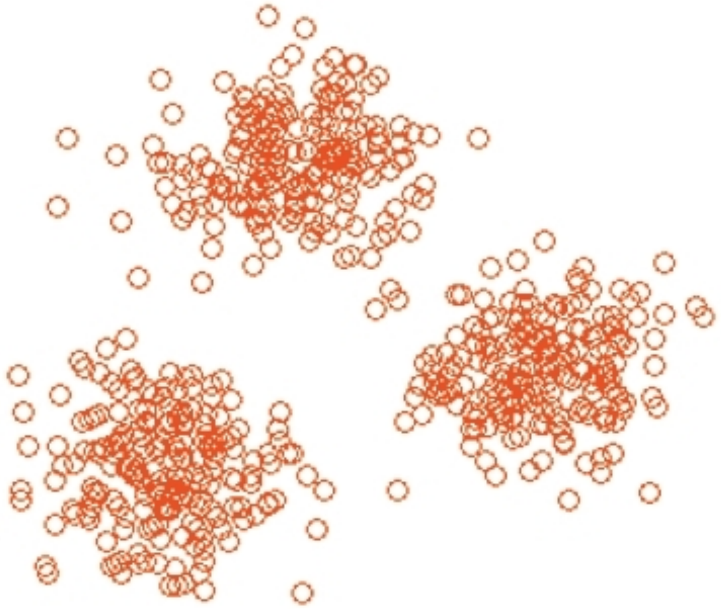
Clustering

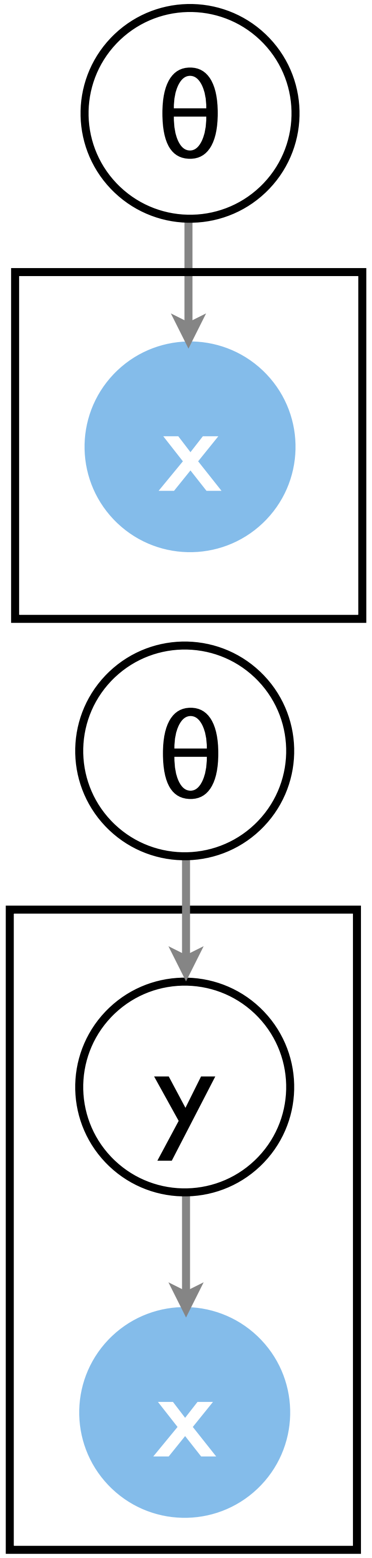
Density Estimation

$$p(x, \theta) = p(\theta) \prod_{i=1}^{n} p(x_i|\theta)$$

Clustering

$$p(x, y, \theta) = p(\pi) \prod_{k=1}^{K} p(\theta_k) \prod_{i=1}^{n} p(y_i|\pi) p(x_i|\theta, y_i)$$





Clustering

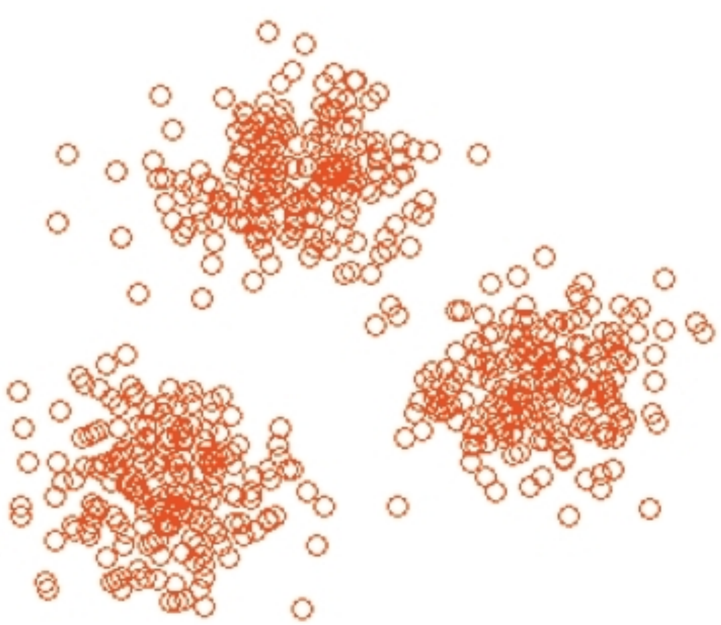
Density Estimation

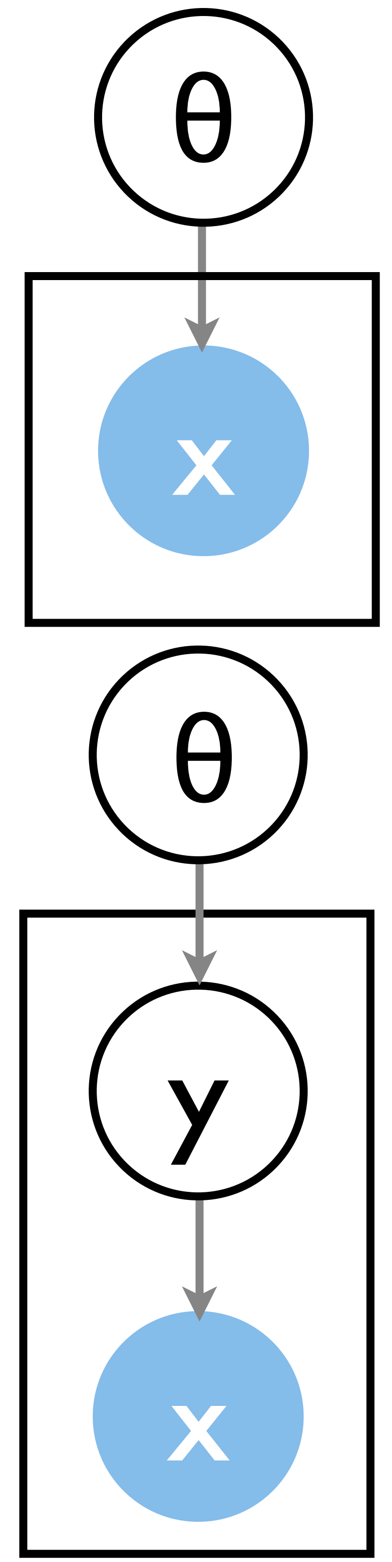
$$p(x,\theta) = p(\theta)\prod_{i=1}^{n} p(x_i|\theta)$$

find θ

Clustering

$$p(x,y,\theta) = p(\pi)\prod_{k=1}^{K} p(\theta_k)\prod_{i=1}^{n} p(y_i|\pi)p(x_i|\theta,y_i)$$





Clustering

Density Estimation

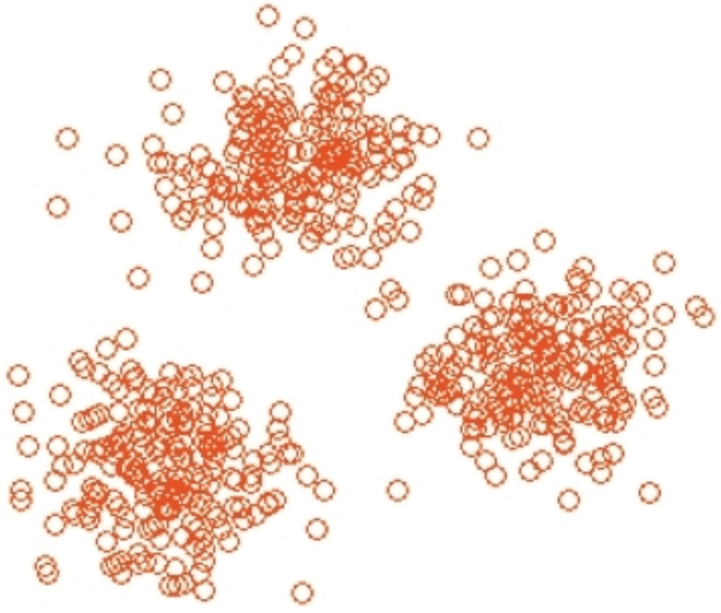
log-concave

$$p(x, \theta) = p(\theta) \prod_{i=1}^{n} p(x_i|\theta)$$

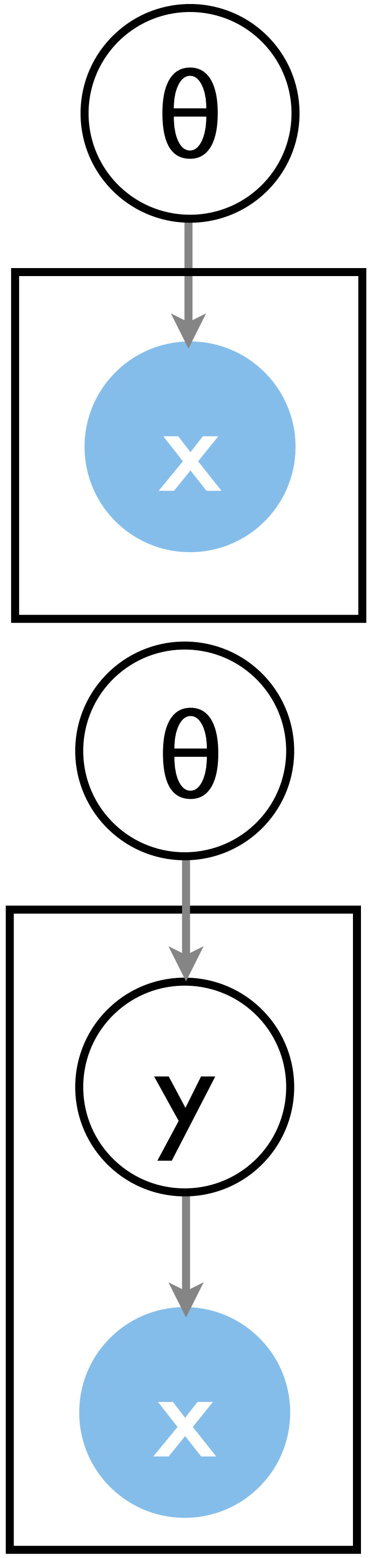
find θ

Clustering

$$p(x, y, \theta) = p(\pi) \prod_{k=1}^{K} p(\theta_k) \prod_{i=1}^{n} p(y_i|\pi) p(x_i|\theta, y_i)$$



general nonlinear



Clustering

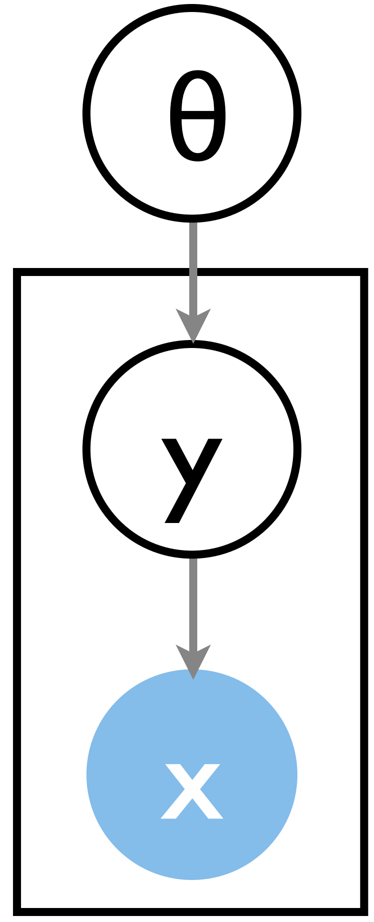
- Optimization problem

$$\underset{\theta}{\text{maximize}} \sum_{y} p(x, y, \theta)$$

$$\underset{\theta}{\text{maximize}} \log p(\pi) + \sum_{k=1}^{K} \log p(\theta_k) + \sum_{i=1}^{n} \log \sum_{y_i \in \mathcal{Y}} \left[p(y_i|\pi) p(x_i|\theta, y_i) \right]$$

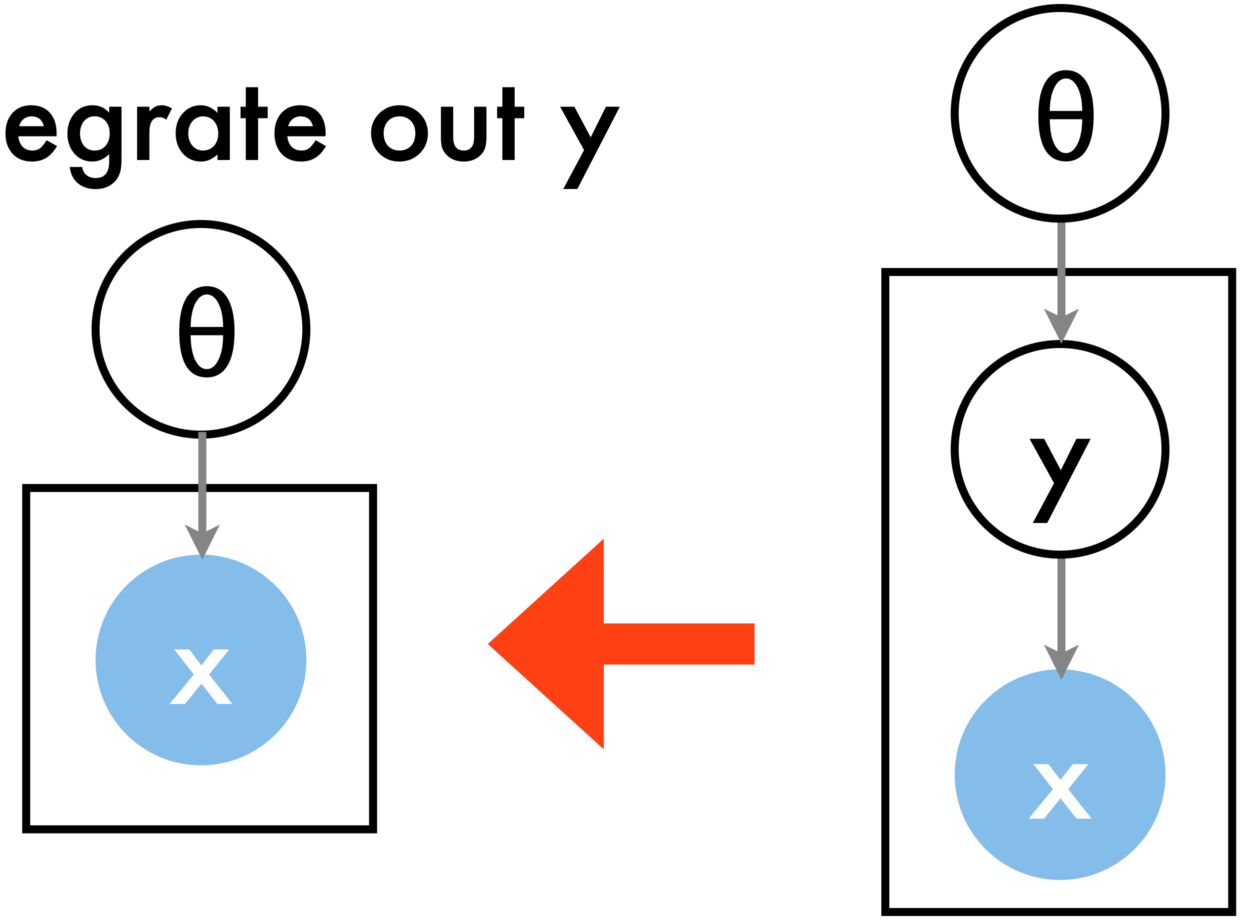
- Options
  - Direct nonconvex optimization (e.g. BFGS)
  - Sampling (draw from the joint distribution) for memory efficiency
  - Variational approximation (concave lower bounds aka EM algorithm)

Clustering



Clustering

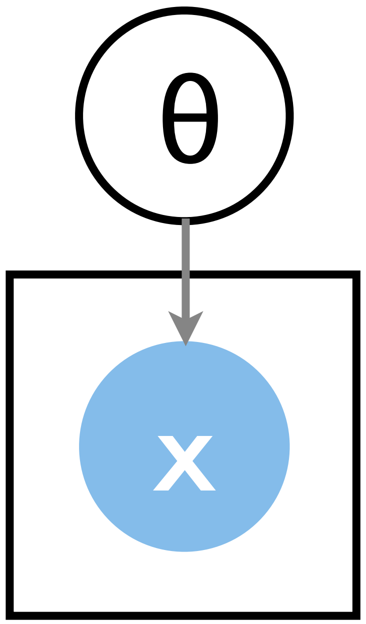
- Integrate out y



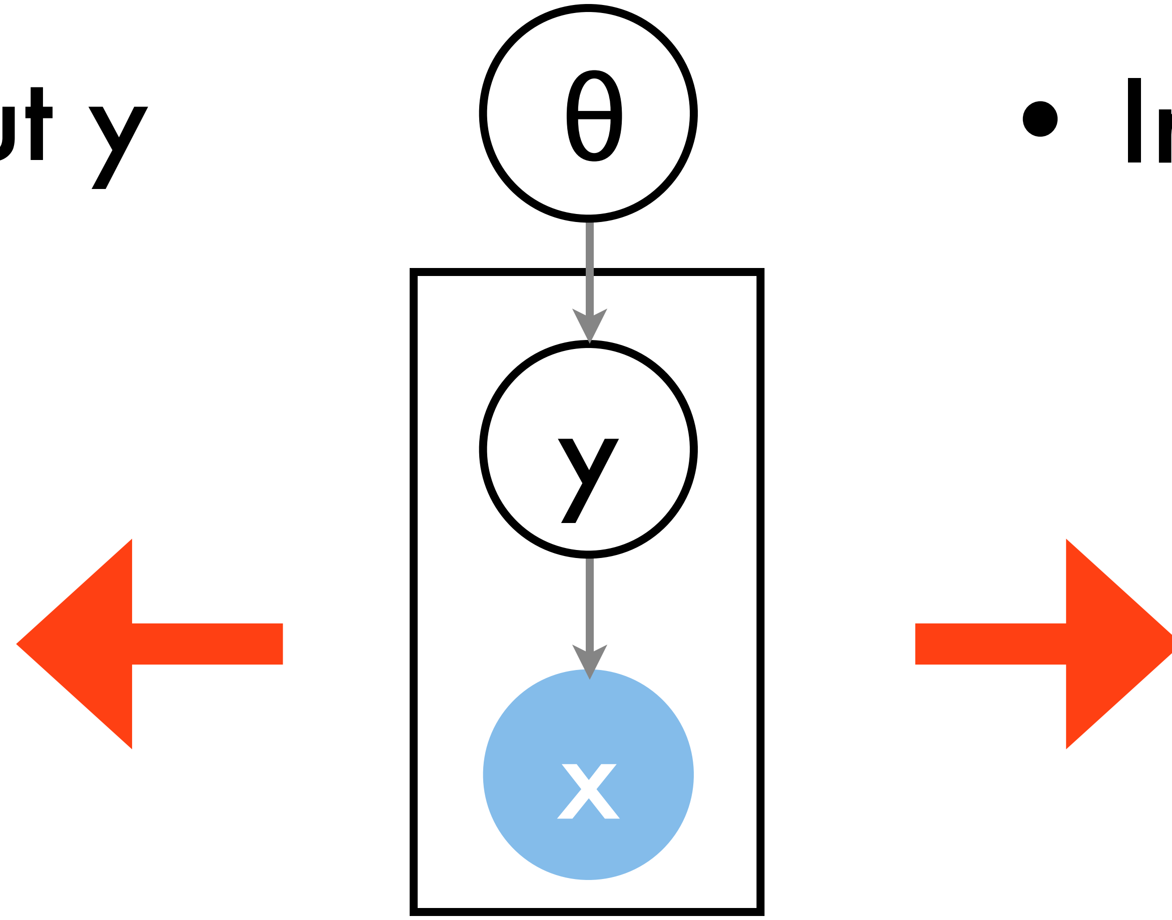
- Nonconvex optimization problem
- EM algorithm

Clustering

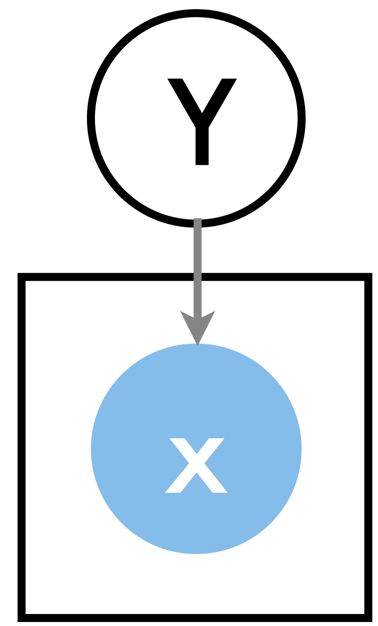
- Integrate out y



- Nonconvex optimization problem
- EM algorithm



- Integrate out θ



- Y is coupled
- Sampling
- Collapsed p

$$p(y|x) \propto p(\{x\} \mid \{x_i : y_i = y\} \cup X_{\text{fake}}) p(y|Y \cup Y_{\text{fake}})$$

Gibbs sampling

- Sampling:
  Draw an instance x from distribution p(x)
- Gibbs sampling:
  - In most cases direct sampling not possible
  - Draw one set of variables at a time

		
	0.45	0.05
	0.05	0.45

Gibbs sampling

- Sampling:
  Draw an instance x from distribution p(x)
- Gibbs sampling:
  - In most cases direct sampling not possible
  - Draw one set of variables at a time

		
	0.45	0.05
	0.05	0.45

(b,g) - draw p(.,g)

Gibbs sampling

- Sampling:
  Draw an instance x from distribution p(x)
- Gibbs sampling:
  - In most cases direct sampling not possible
  - Draw one set of variables at a time

		
	0.45	0.05
	0.05	0.45

(b,g) - draw p(.,g)
(g,g) - draw p(g,.)

Gibbs sampling

- Sampling:
  Draw an instance x from distribution p(x)
- Gibbs sampling:
  - In most cases direct sampling not possible
  - Draw one set of variables at a time

		
	0.45	0.05
	0.05	0.45

(b,g) - draw p(.,g)
(g,g) - draw p(g,.)
(g,g) - draw p(.,g)

Gibbs sampling

- Sampling:
  Draw an instance x from distribution p(x)
- Gibbs sampling:
  - In most cases direct sampling not possible
  - Draw one set of variables at a time

		
	0.45	0.05
	0.05	0.45

(b,g) - draw p(.,g)
(g,g) - draw p(g,.)
(g,g) - draw p(.,g)
(b,g) - draw p(b,.)

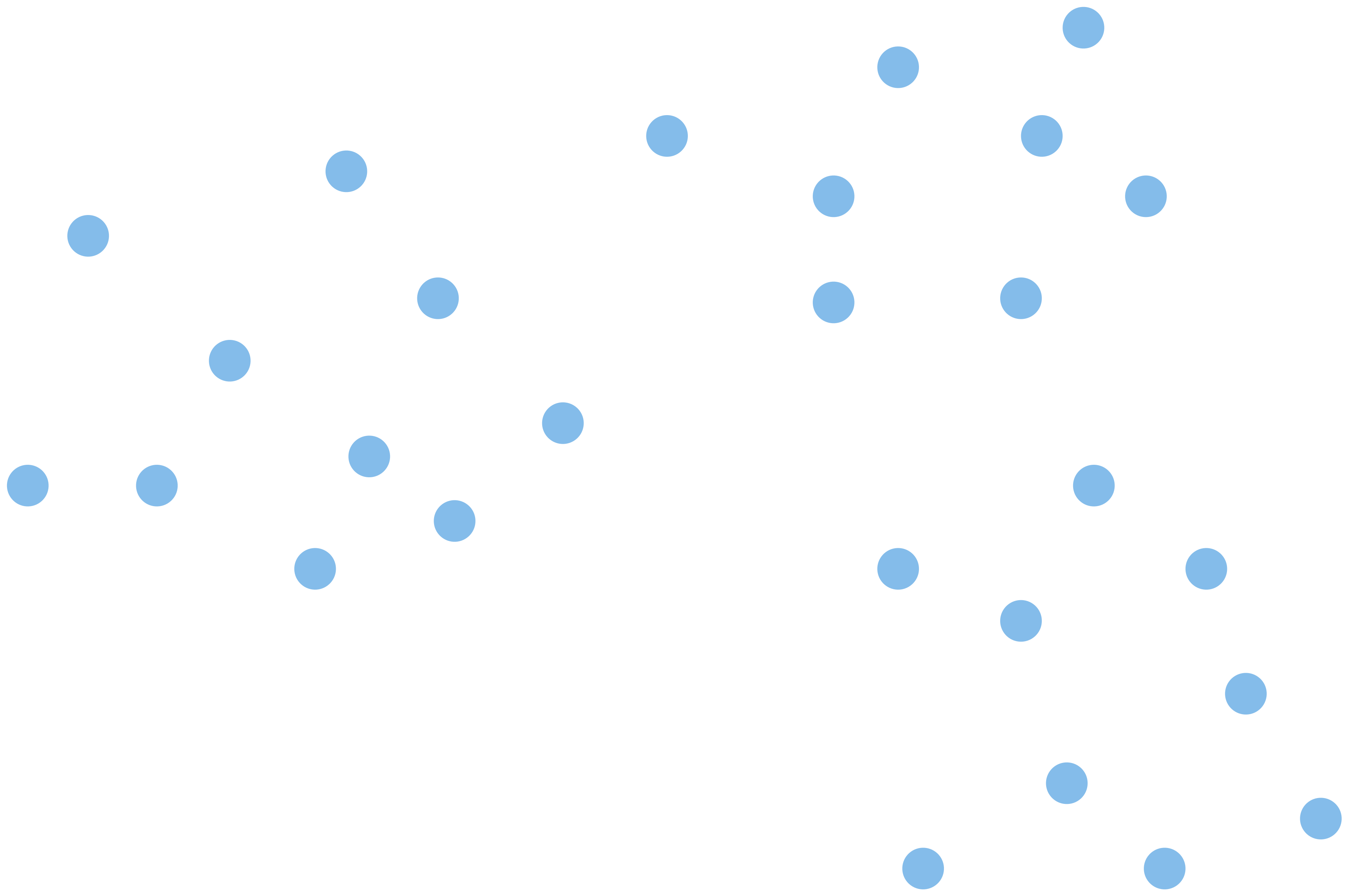
Gibbs sampling

- Sampling:
  Draw an instance x from distribution p(x)
- Gibbs sampling:
  - In most cases direct sampling not possible
  - Draw one set of variables at a time

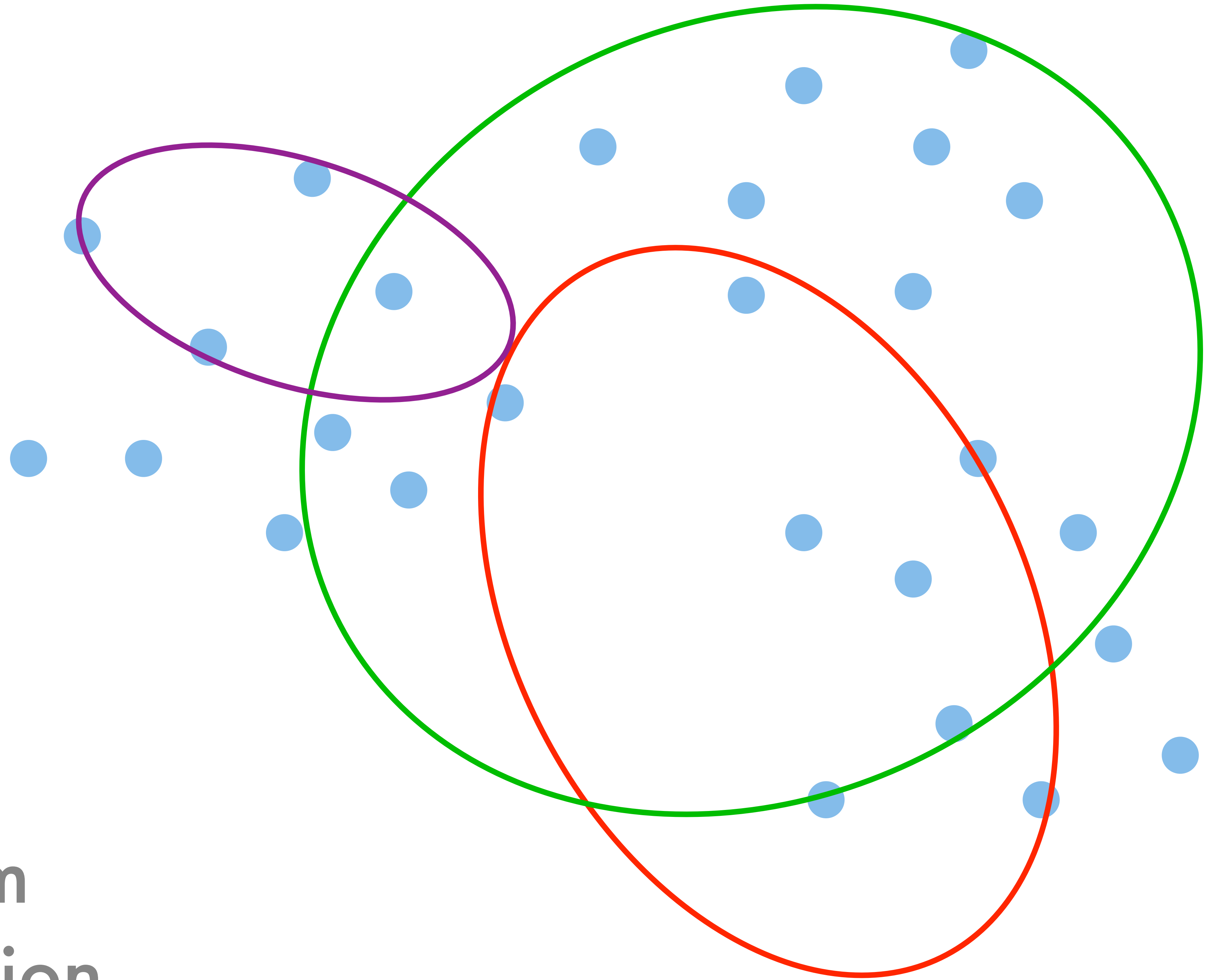
		
	0.45	0.05
	0.05	0.45

(b,g) - draw p(.,g)
(g,g) - draw p(g,.)
(g,g) - draw p(.,g)
(b,g) - draw p(b,.)
(b,b) ...

Gibbs sampling for clustering

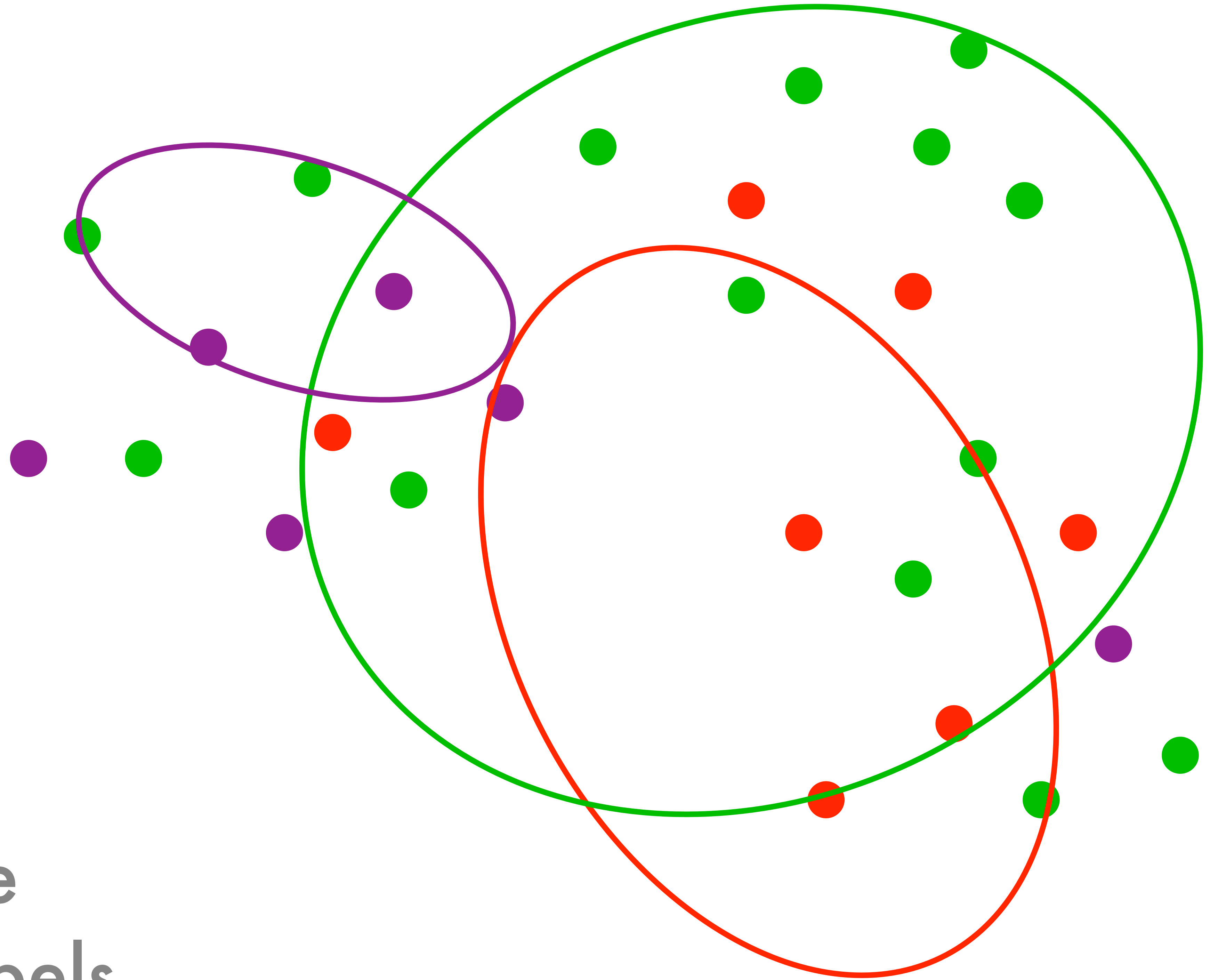


Gibbs sampling for clustering



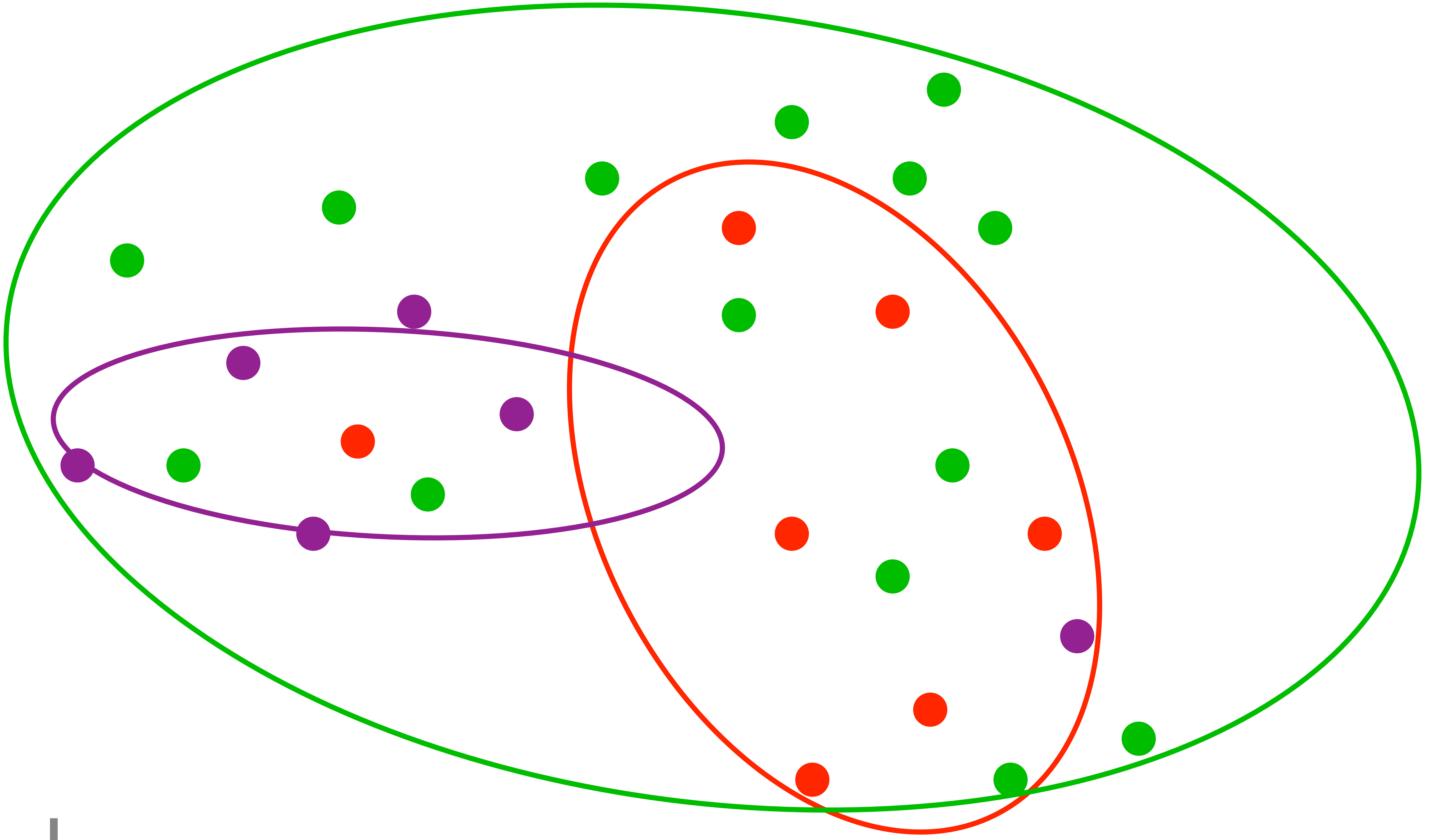
random
initialization

Gibbs sampling for clustering



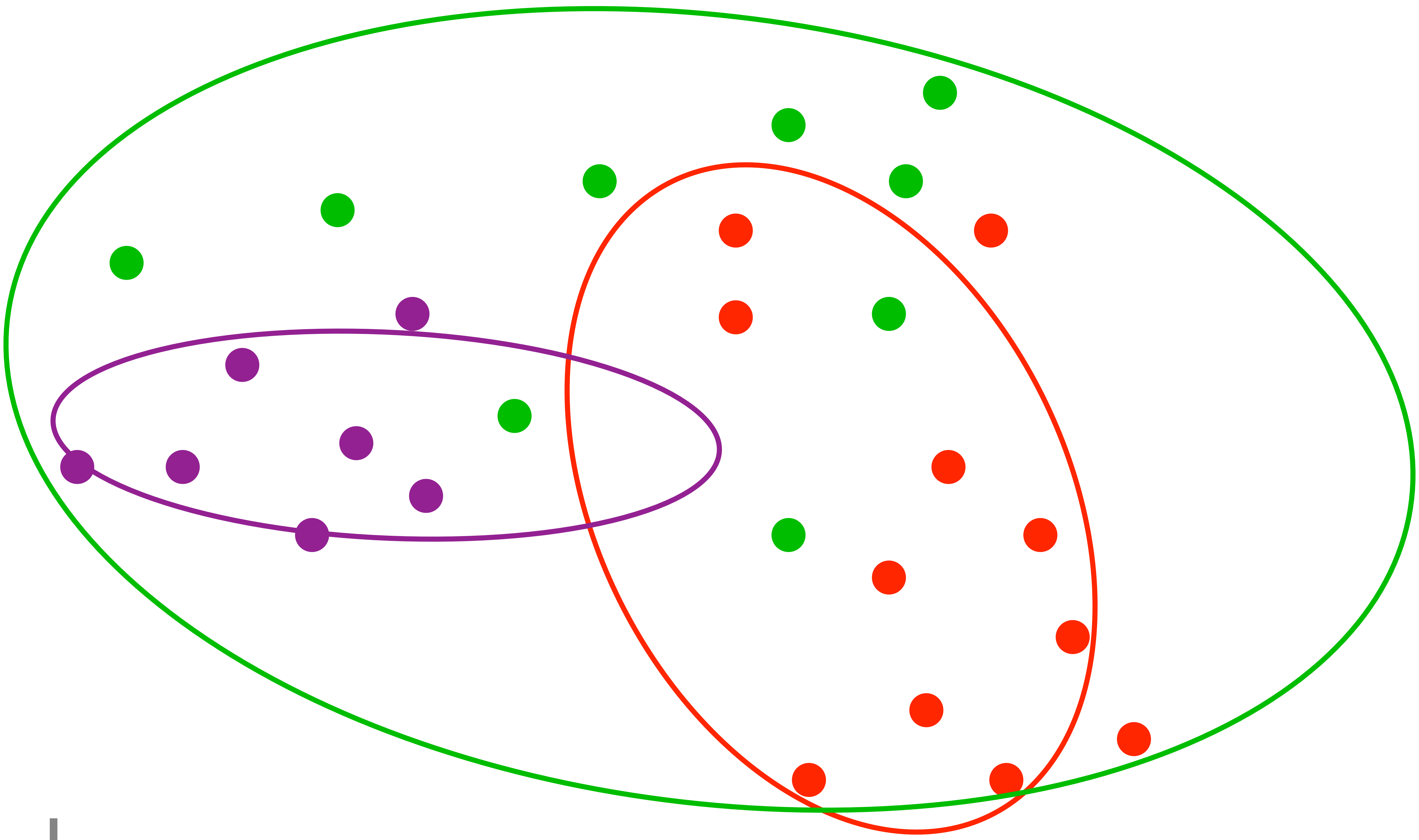
sample
cluster labels

Gibbs sampling for clustering



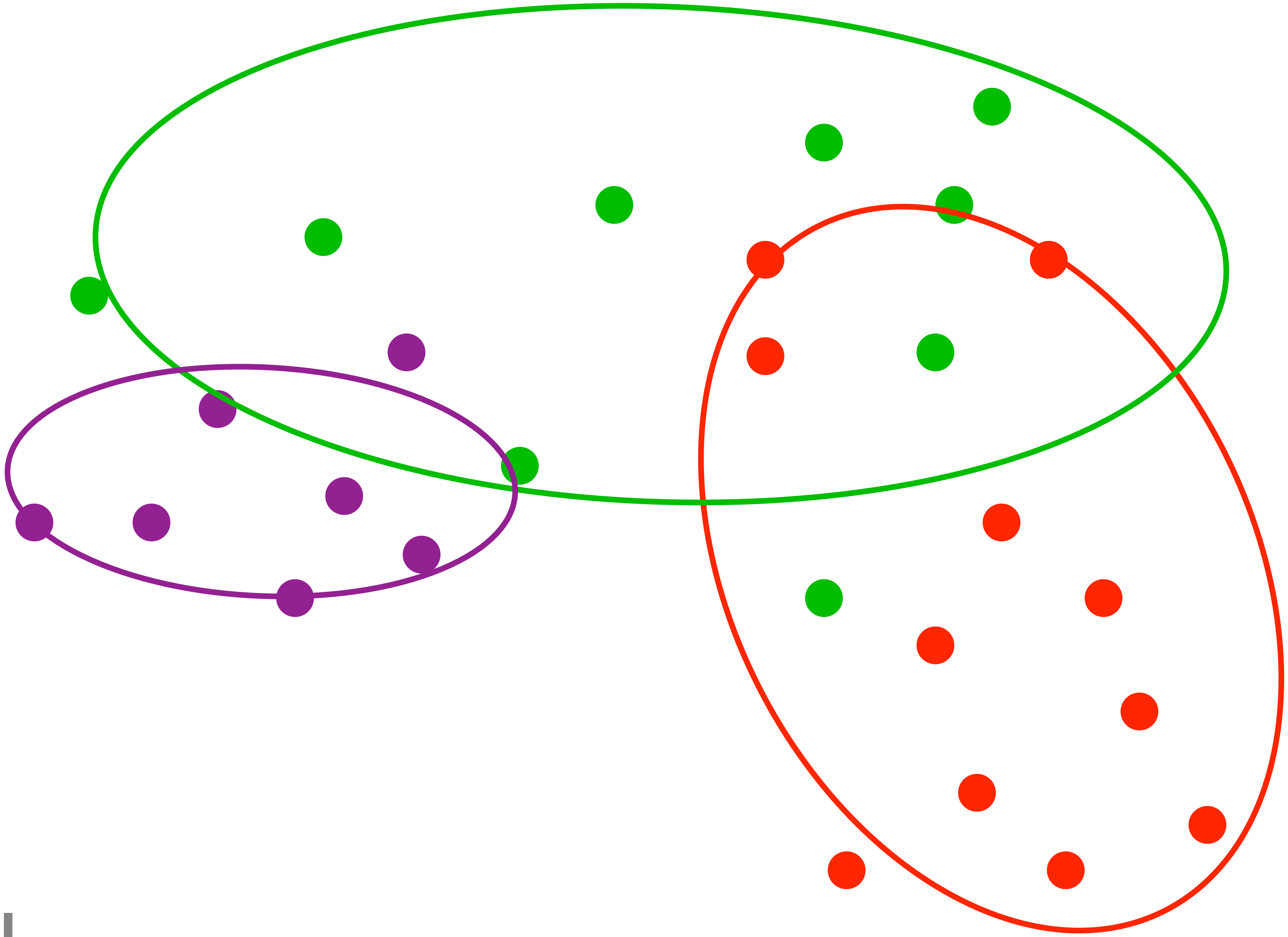
resample
cluster model

Gibbs sampling for clustering



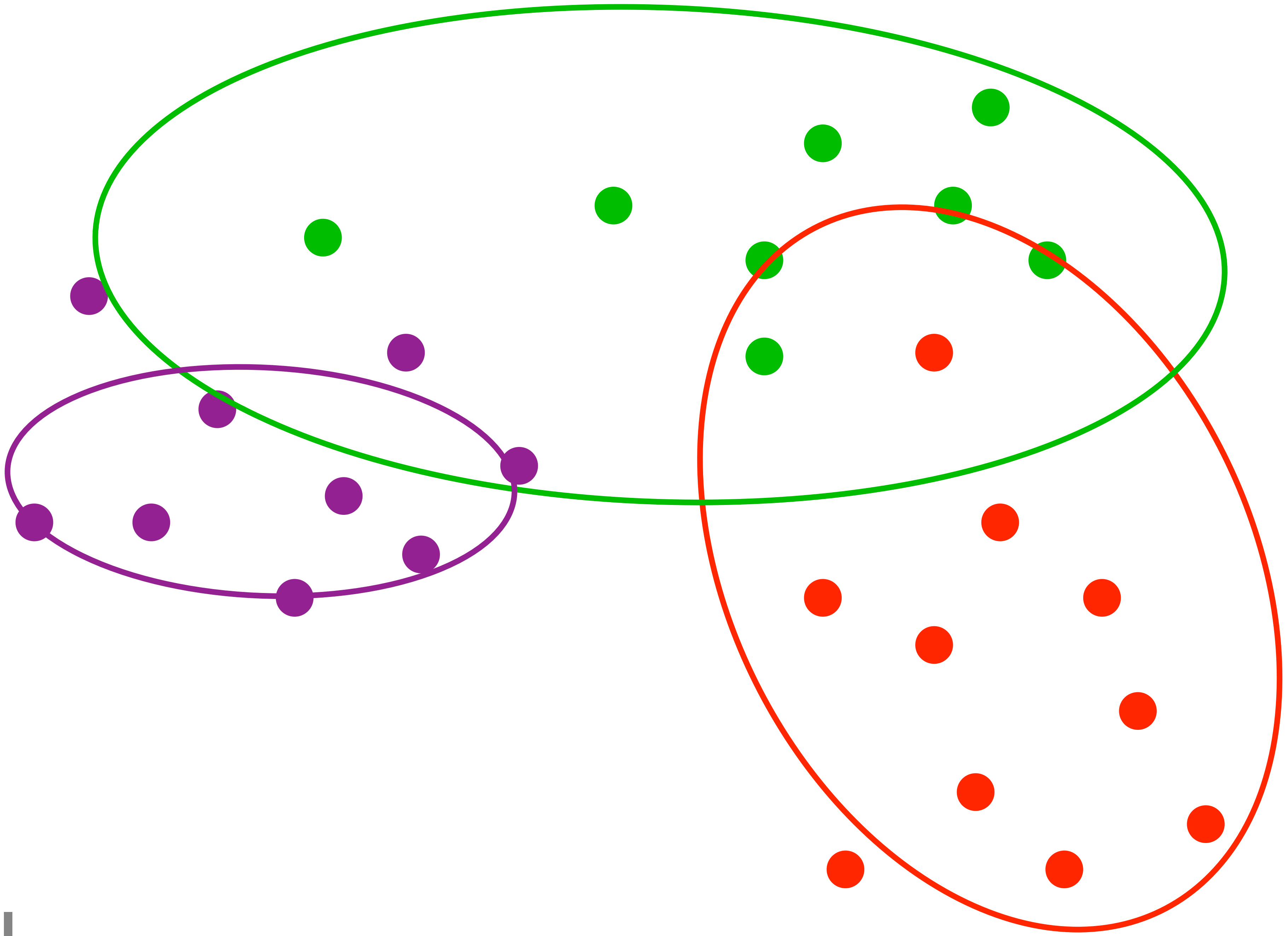
resample
cluster labels

Gibbs sampling for clustering



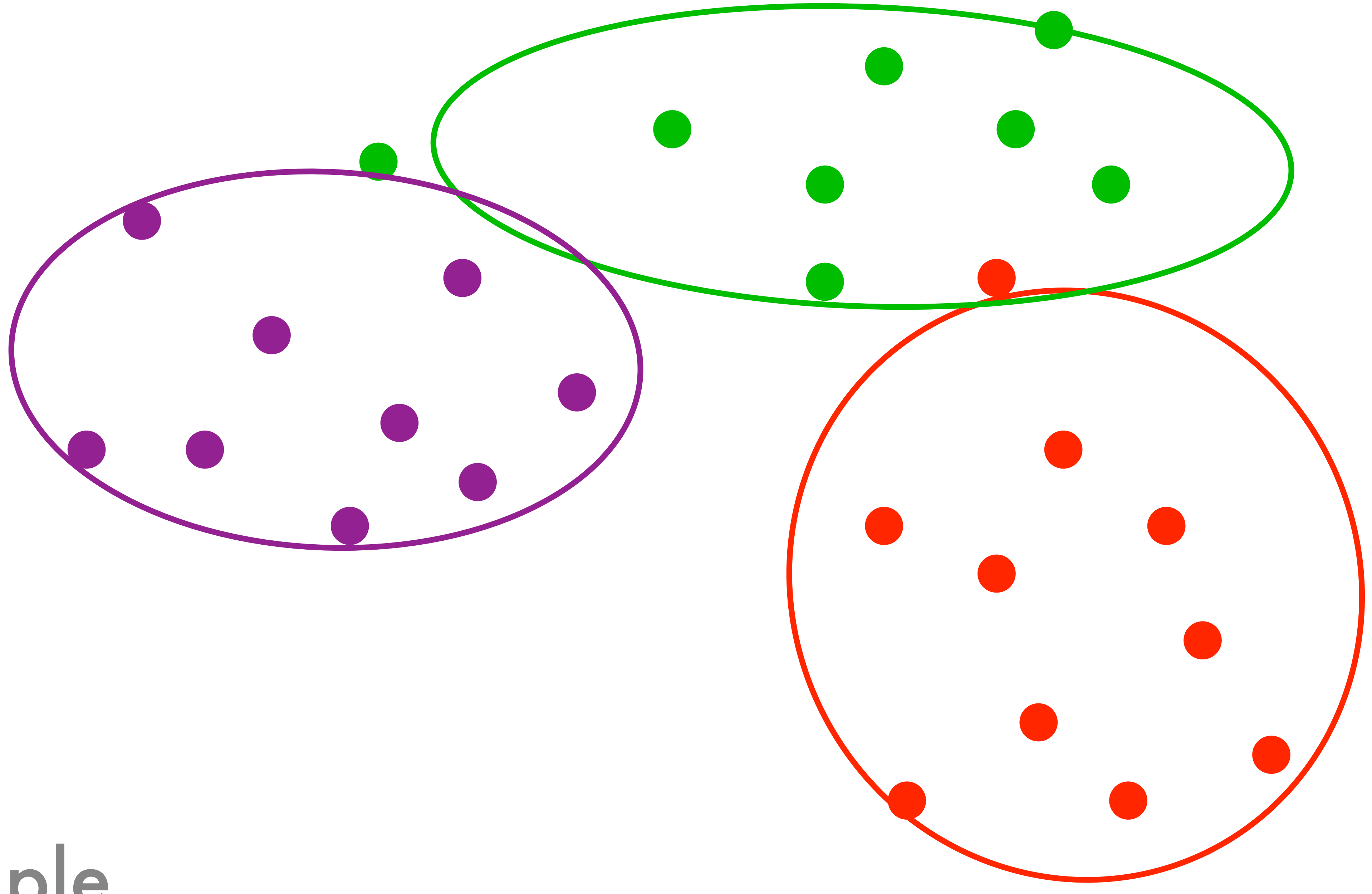
resample
cluster model

Gibbs sampling for clustering



resample
cluster labels

Gibbs sampling for clustering



resample
cluster model

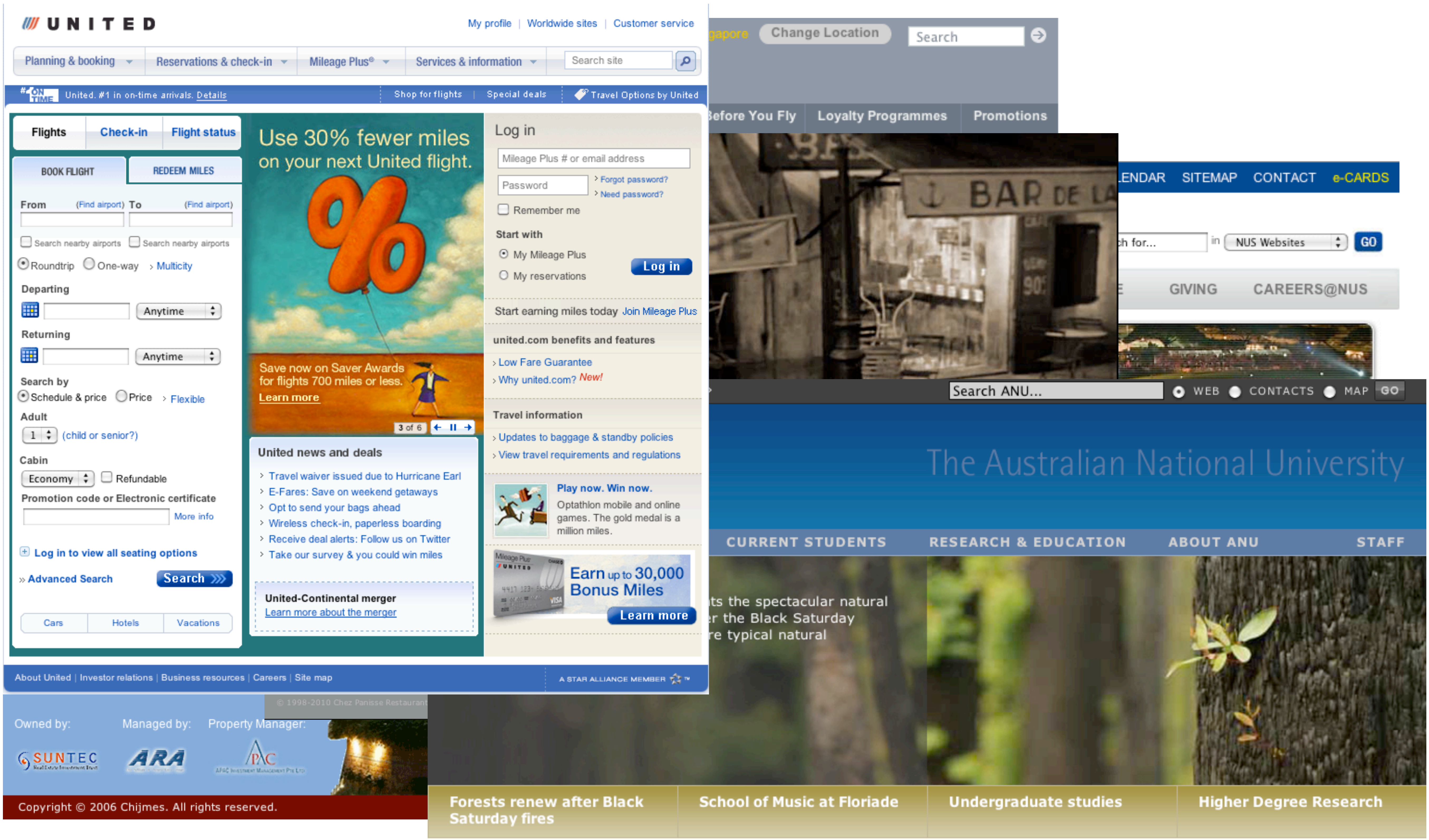
e.g. Mahout Dirichlet Process Clustering

Topic models

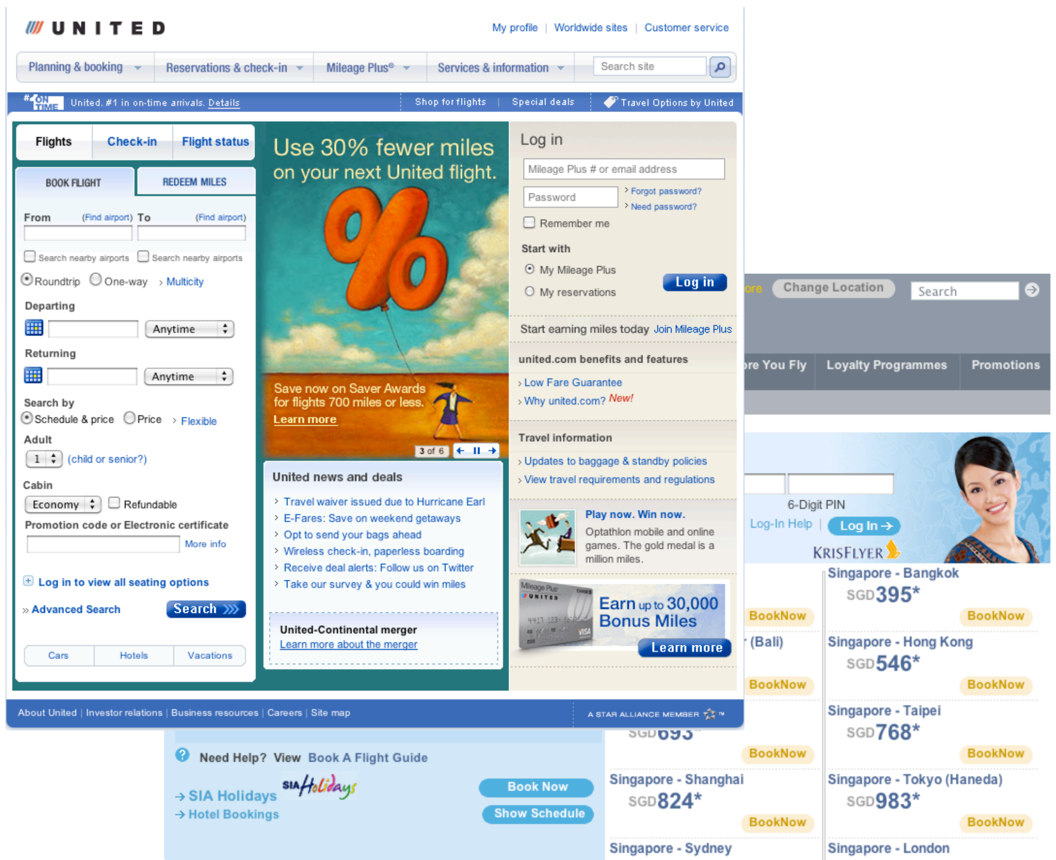
Grouping objects

Grouping objects

Grouping objects

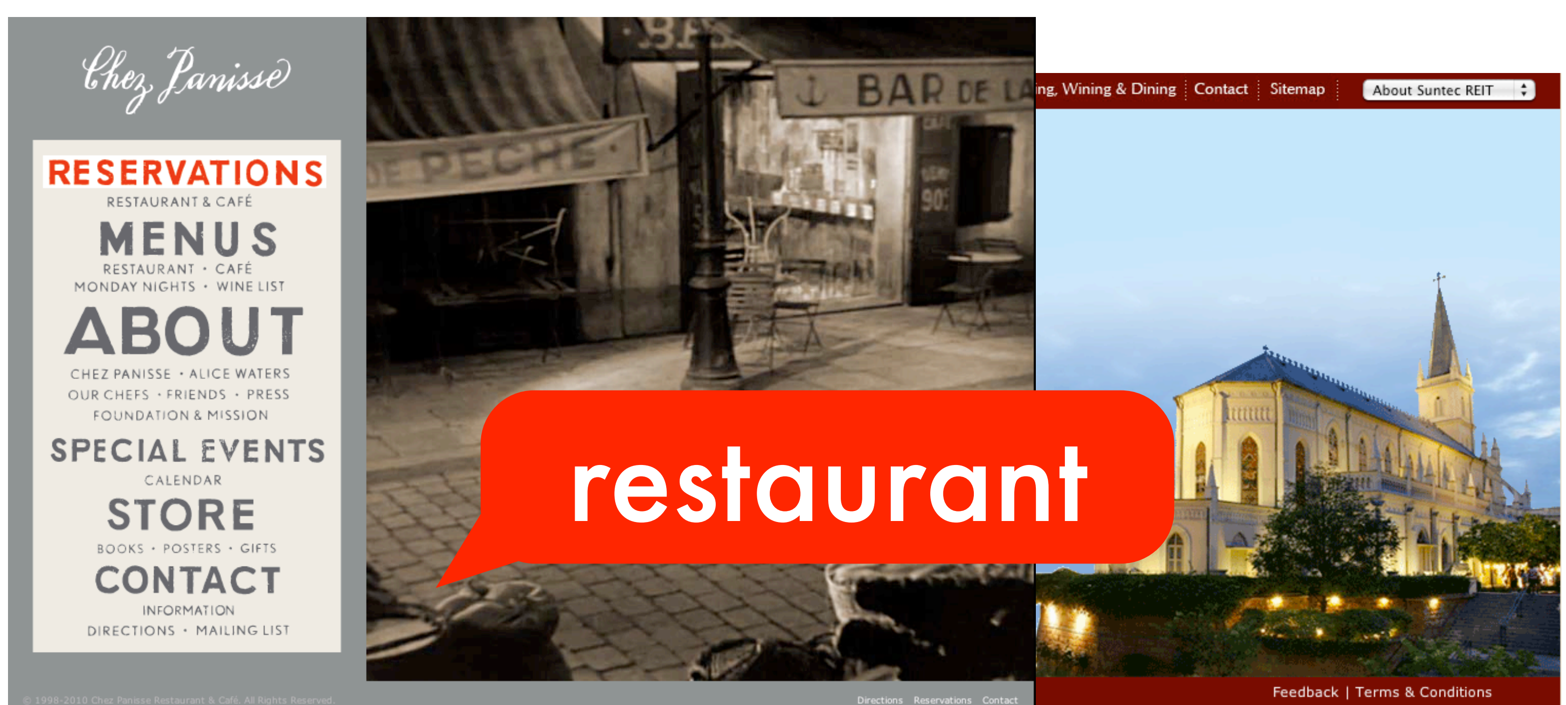


Grouping objects



YAHOO!

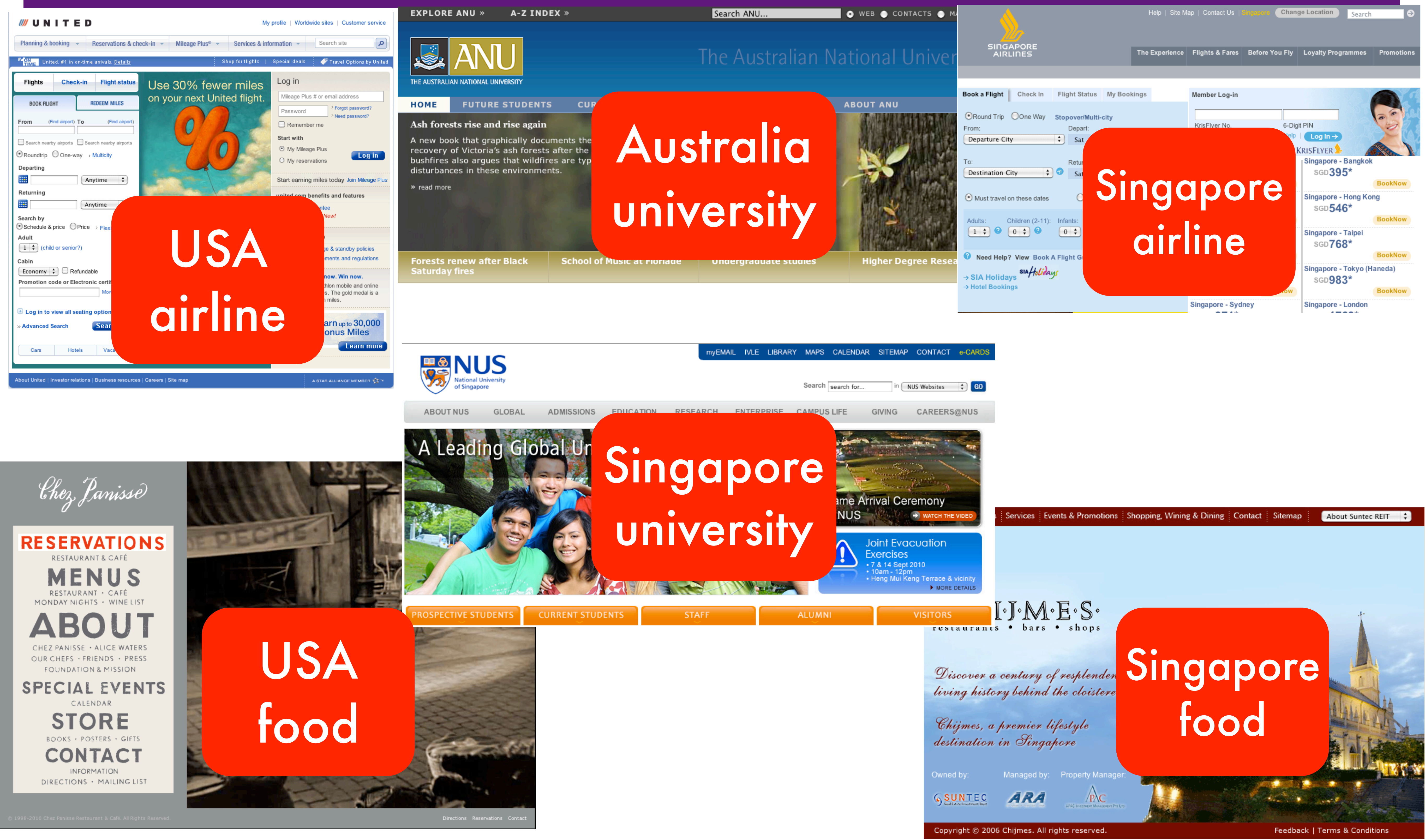
Grouping objects



YAHOO!

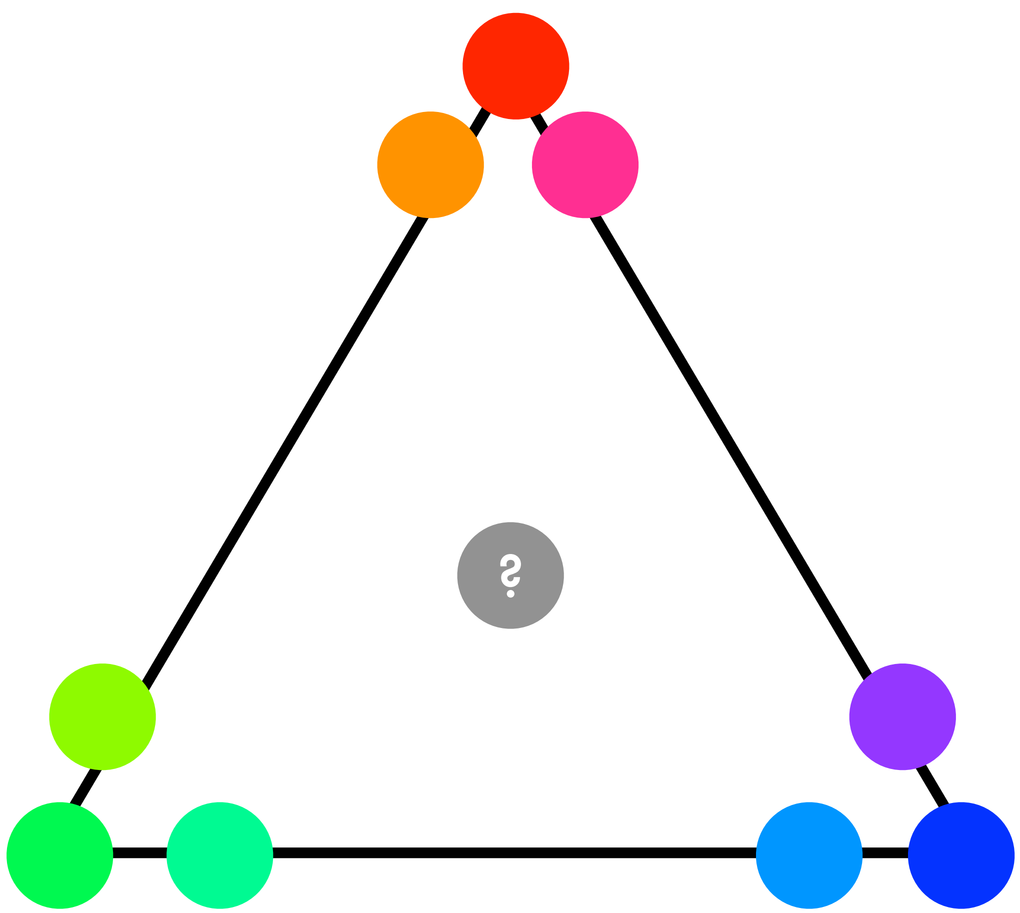
Grouping objects

Topic Models



Clustering & Topic Models

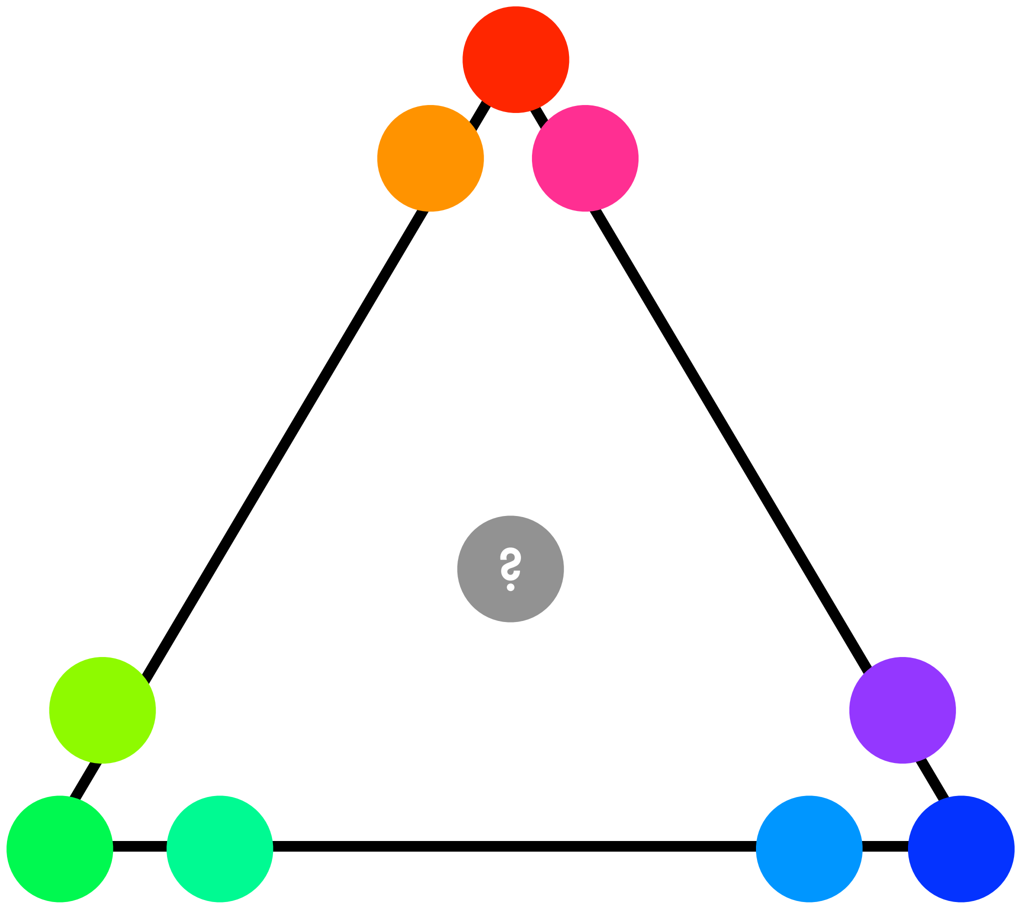
Clustering



group objects
by prototypes

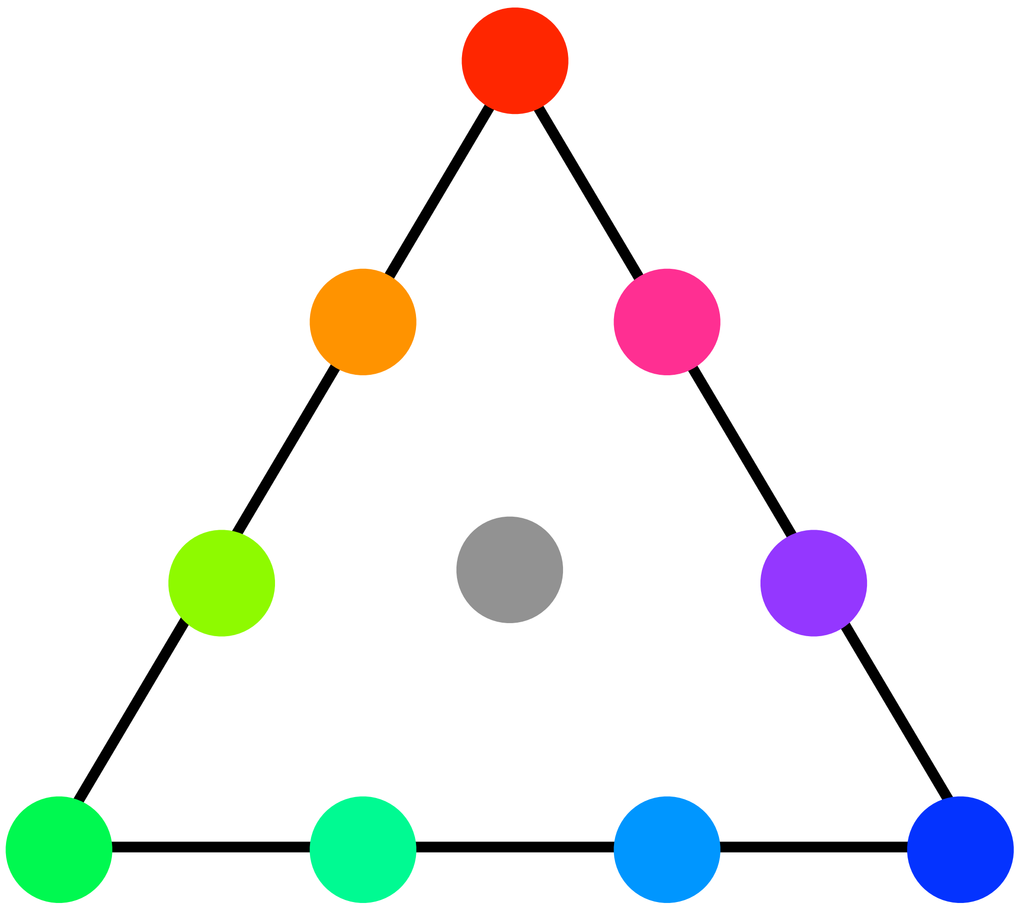
Clustering & Topic Models

Clustering



group objects
by prototypes

Topics

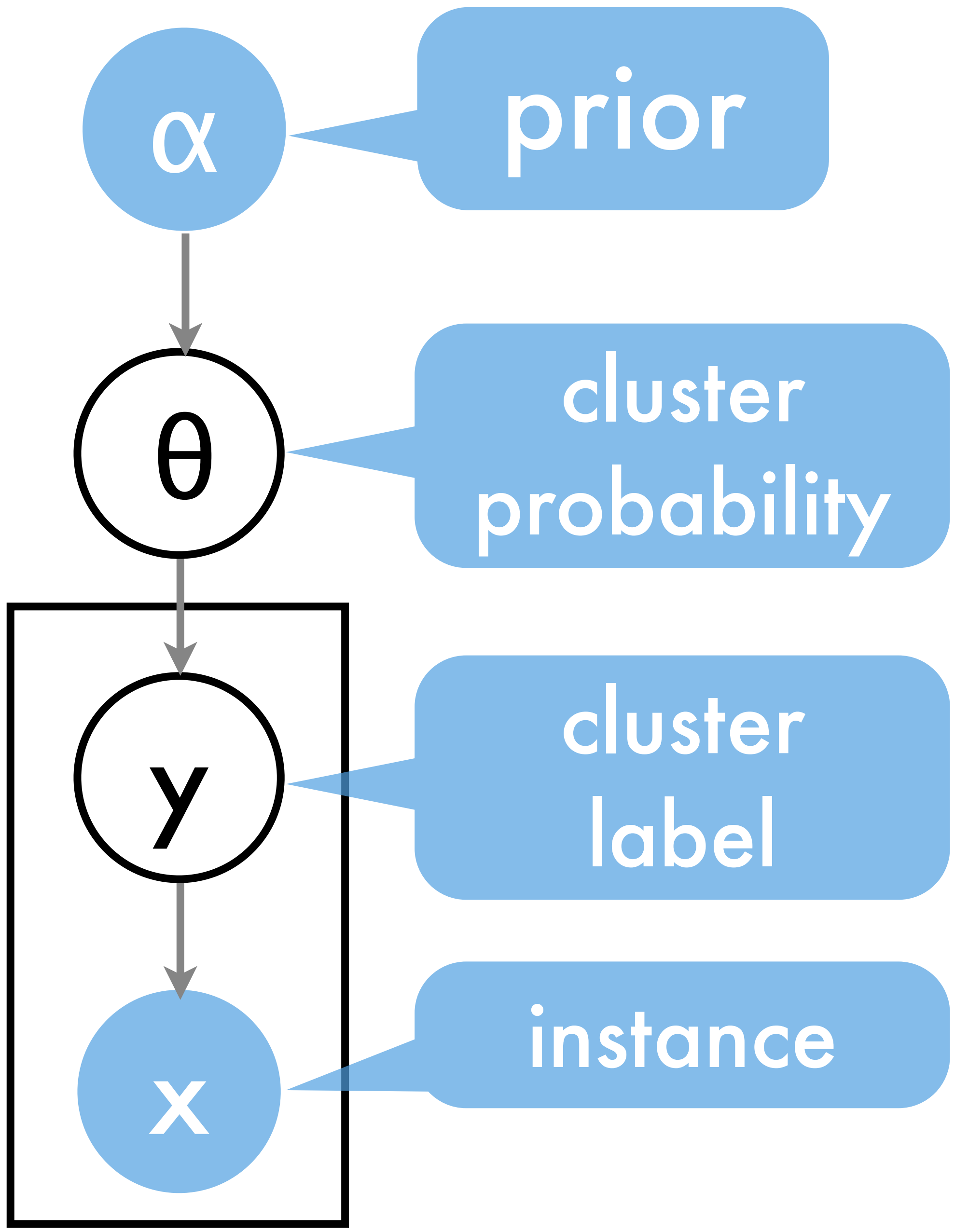


decompose objects
into prototypes

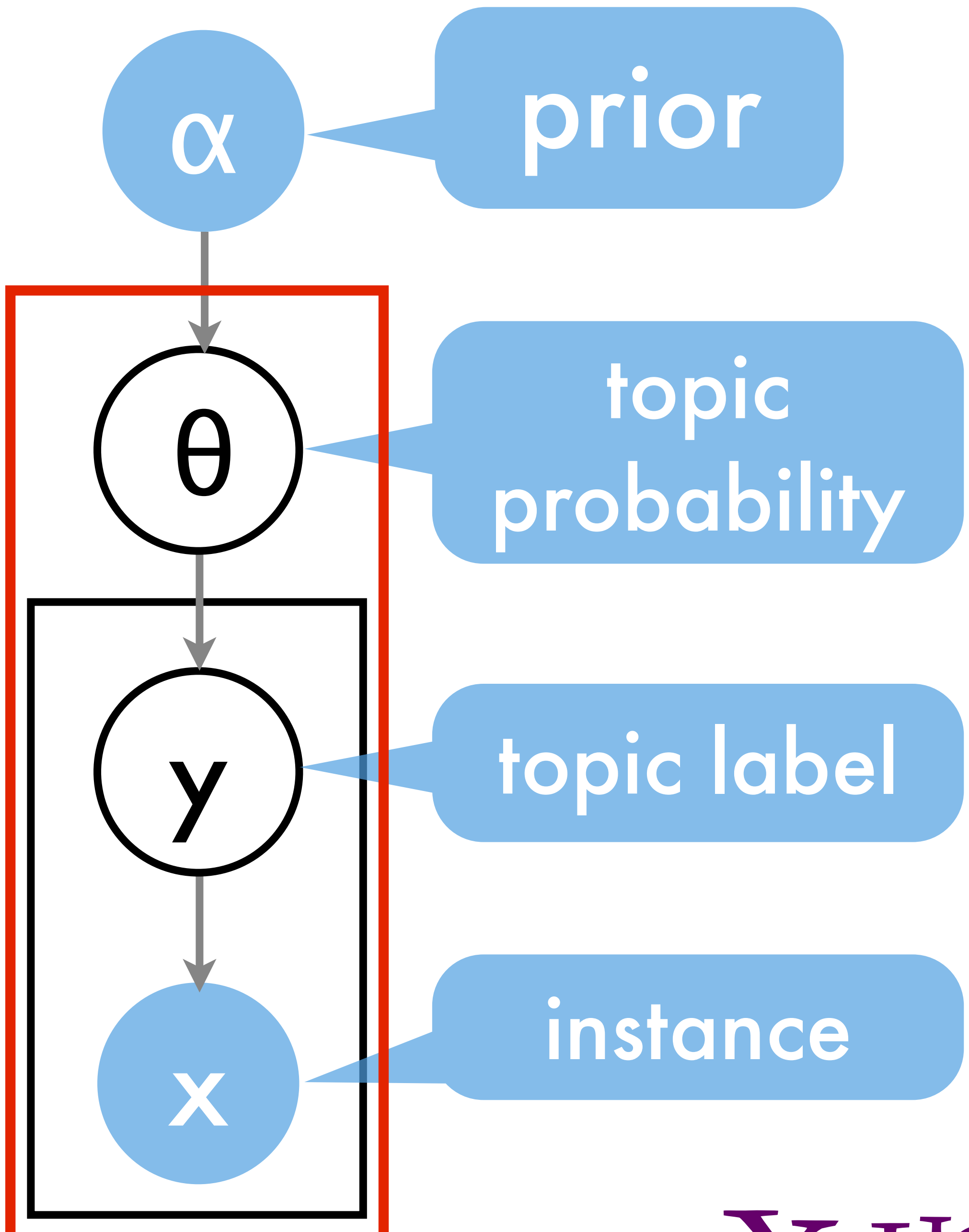
YAHOO!

Clustering & Topic Models

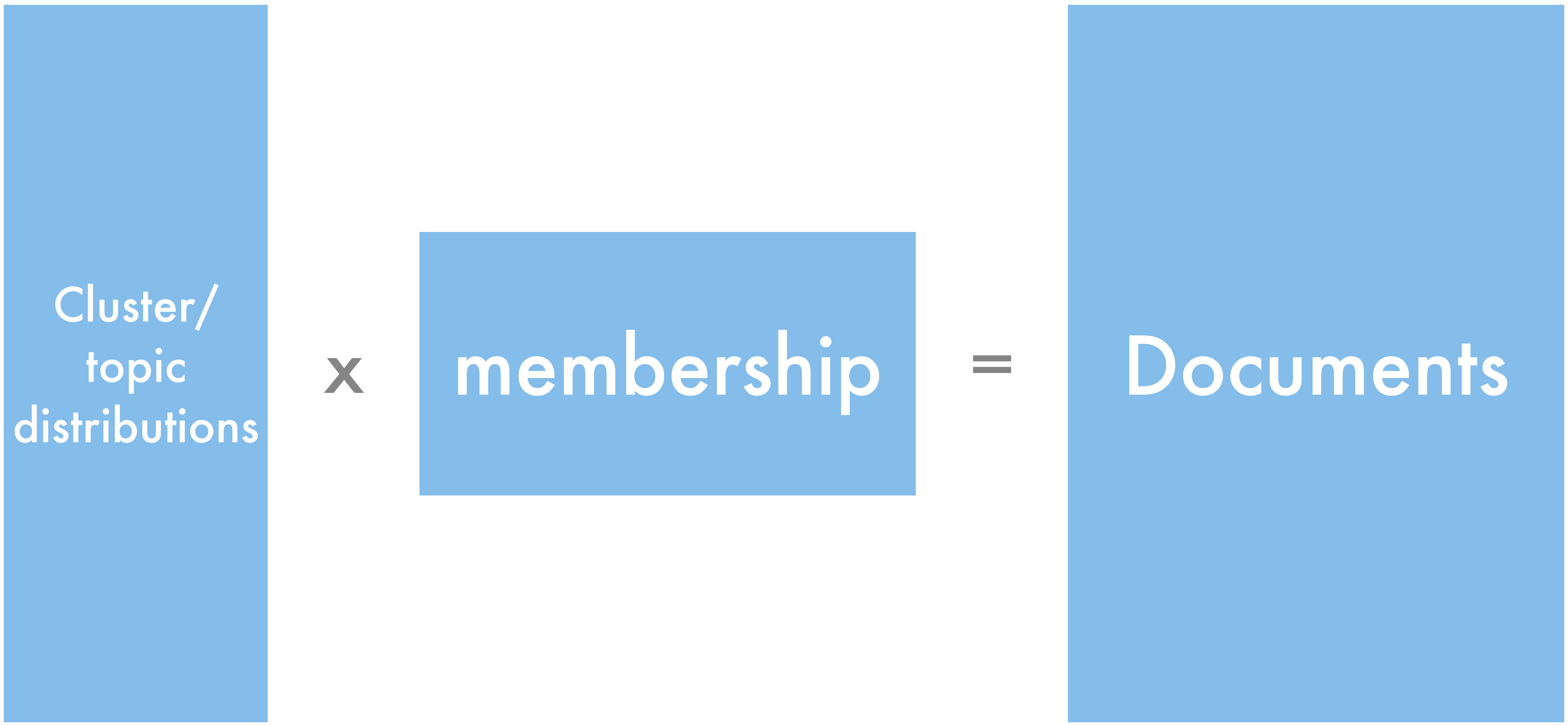
clustering



Latent Dirichlet Allocation



Clustering & Topic Models



clustering: (0, 1) matrix
topic model: stochastic matrix
LSI: arbitrary matrices

Topics in text

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.

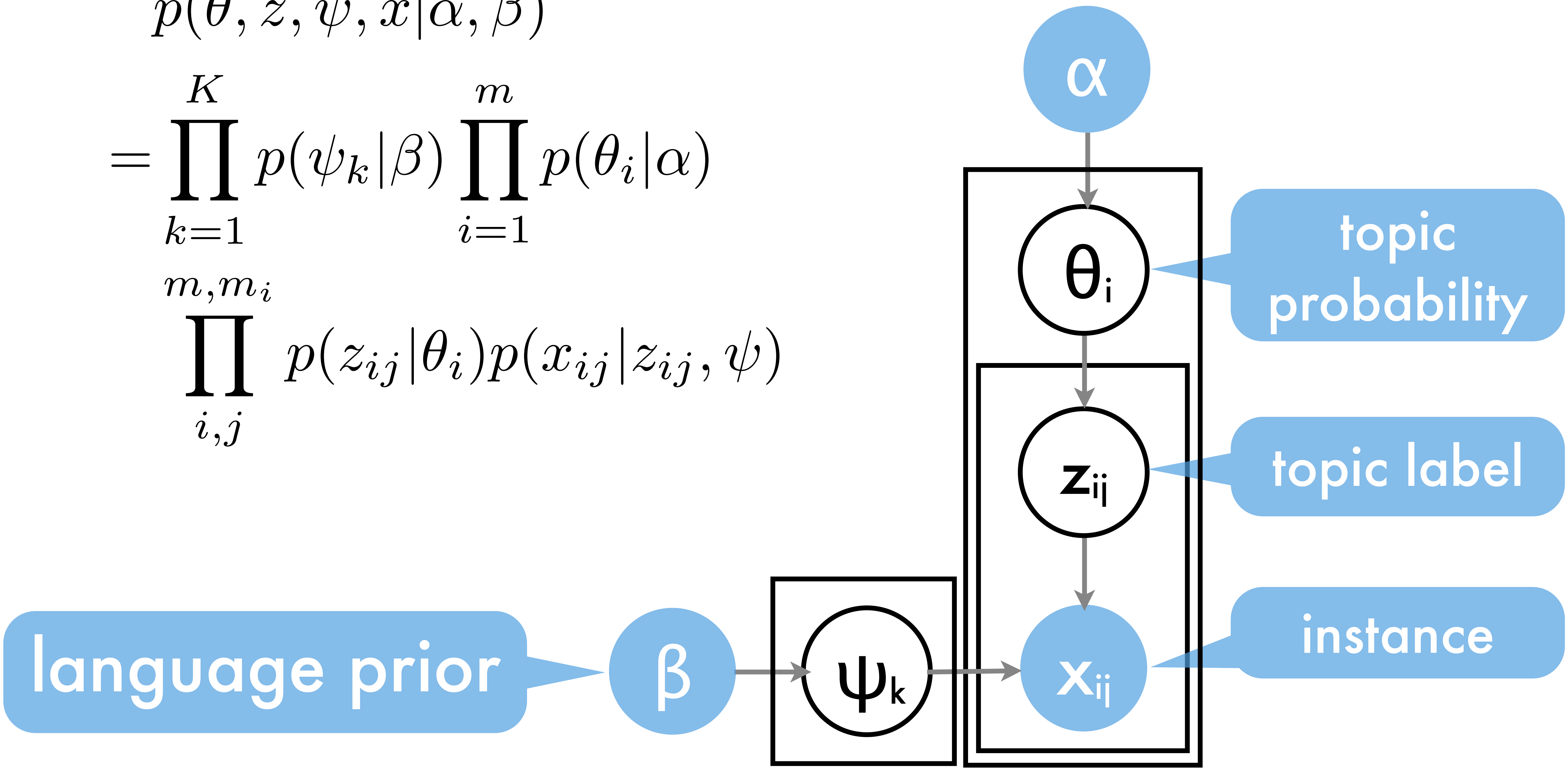
Latent Dirichlet Allocation; Blei, Ng, Jordan, JMLR 2003

Joint Probability Distribution

$$p(\theta, z, \psi, x|\alpha, \beta)$$

$$= \prod_{k=1}^{K} p(\psi_k|\beta) \prod_{i=1}^{m} p(\theta_i|\alpha)$$

$$\prod_{i,j}^{m,m_i} p(z_{ij}|\theta_i)p(x_{ij}|z_{ij}, \psi)$$



Joint Probability Distribution

sample Ψ independently

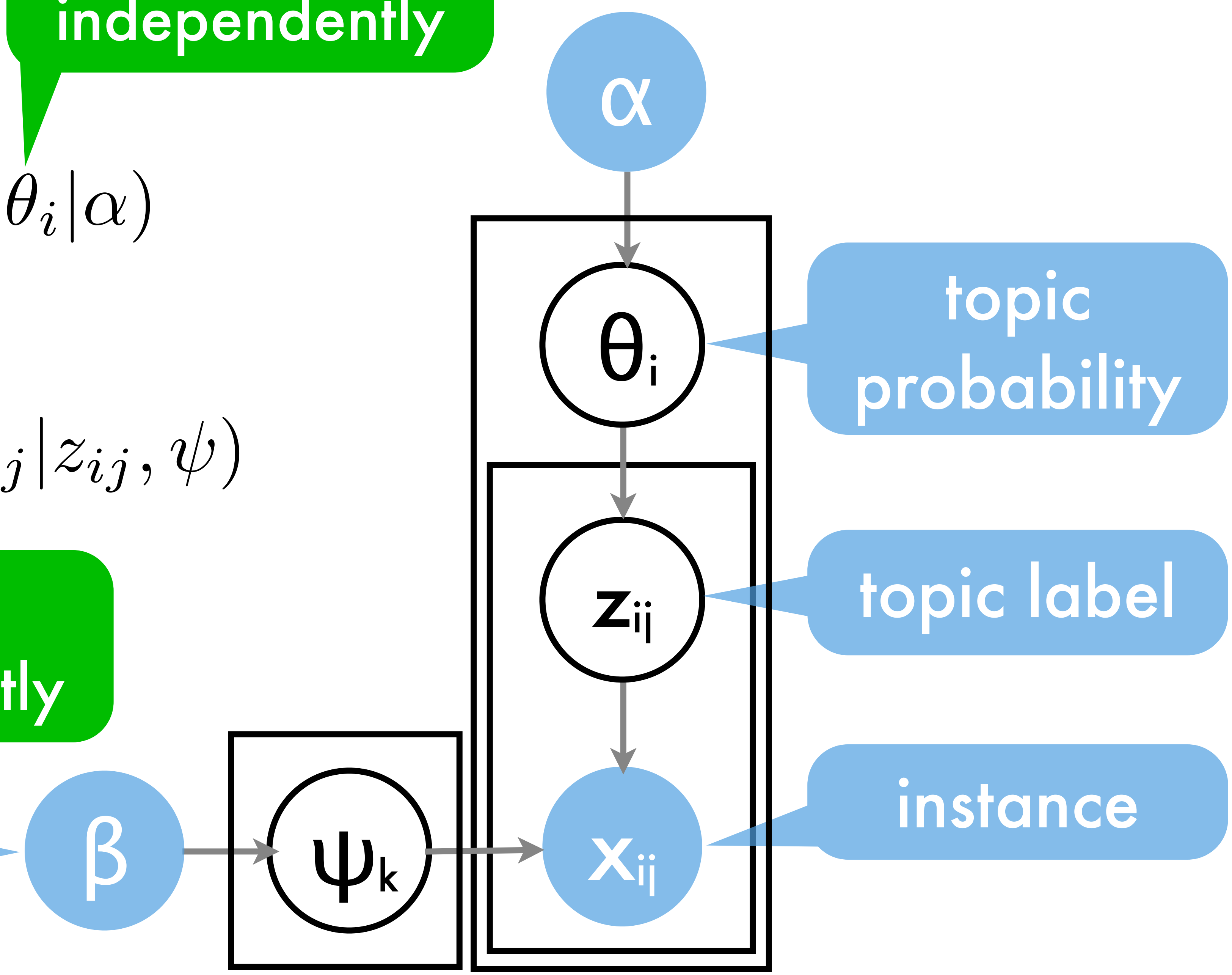
sample θ independently

$$p(\theta, z, \psi, x|\alpha, \beta)$$

$$= \prod_{k=1}^{K} p(\psi_k|\beta) \prod_{i=1}^{m} p(\theta_i|\alpha)$$

$$\prod_{i,j}^{m,m_i} p(z_{ij}|\theta_i) p(x_{ij}|z_{ij}, \psi)$$

sample z independently



Joint Probability Distribution

sample Ψ independently

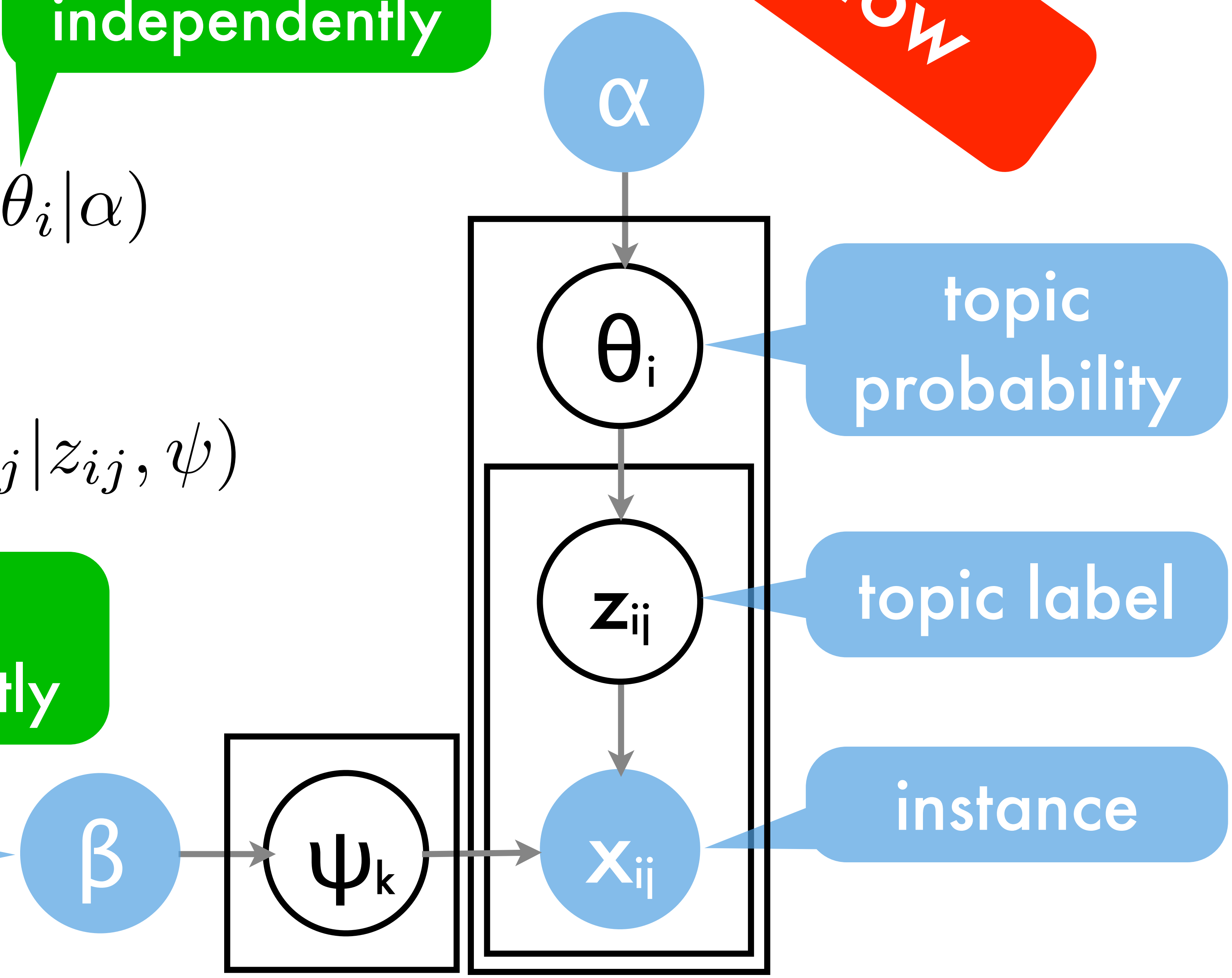
sample θ independently

slow

$$p(\theta, z, \psi, x|\alpha, \beta) = \prod_{k=1}^{K} p(\psi_k|\beta) \prod_{i=1}^{m} p(\theta_i|\alpha) \prod_{i,j}^{m,m_i} p(z_{ij}|\theta_i) p(x_{ij}|z_{ij}, \psi)$$

sample z independently

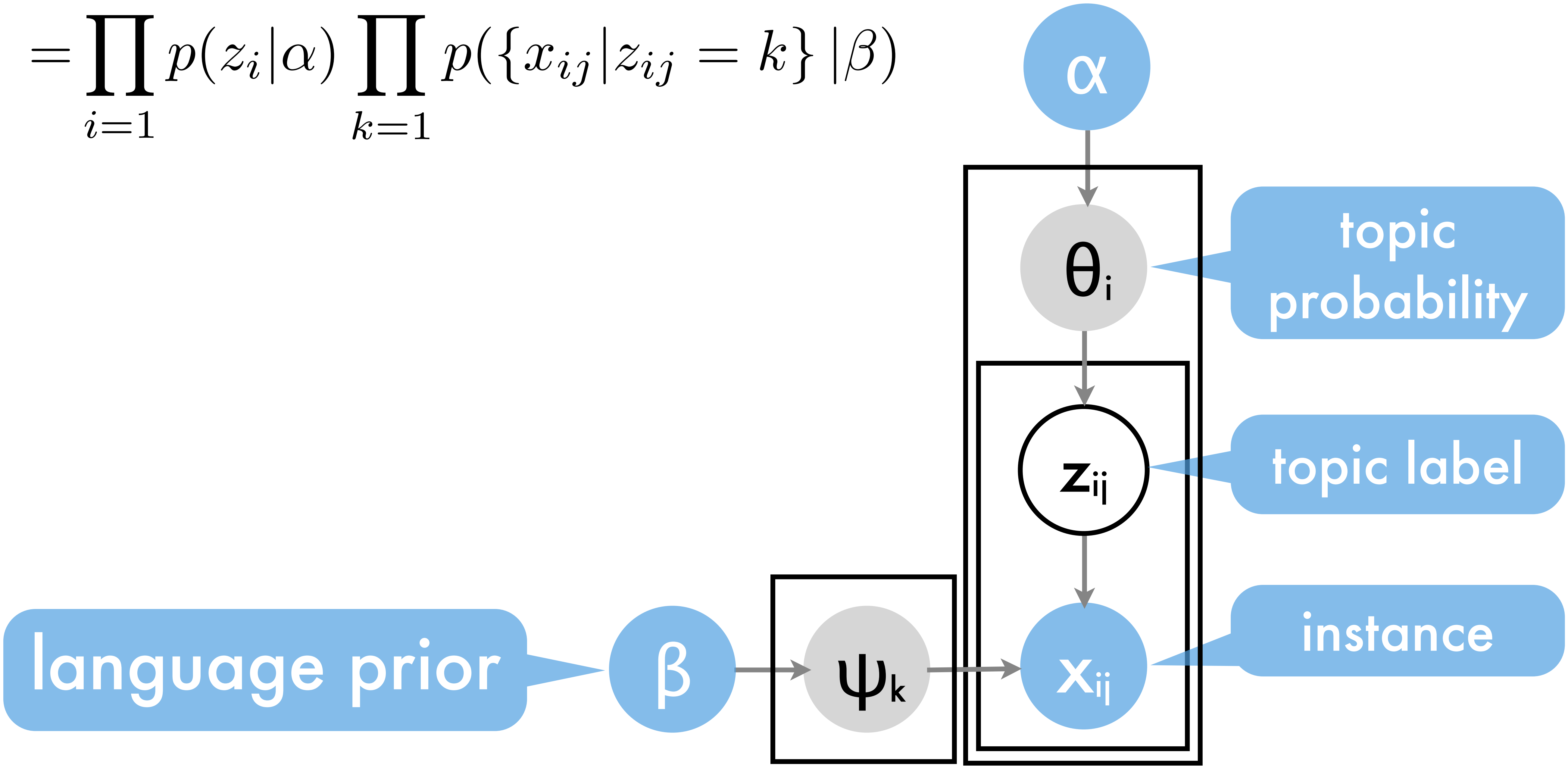
language prior



Collapsed Sampler

$$p(z, x|\alpha, \beta)$$

$$= \prod_{i=1}^{m} p(z_i|\alpha) \prod_{k=1}^{k} p(\{x_{ij}|z_{ij} = k\}|\beta)$$

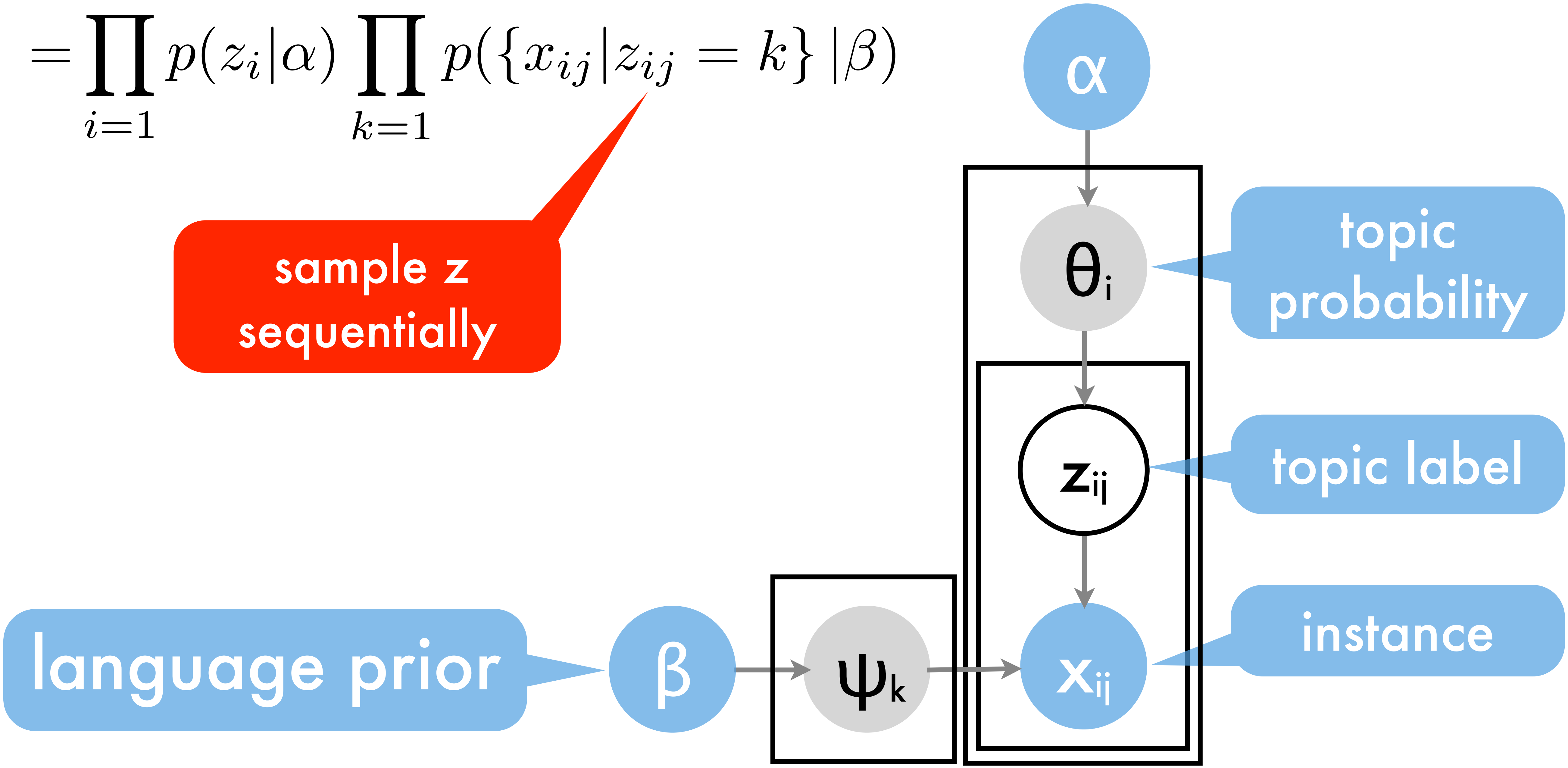


Collapsed Sampler

$$p(z, x|\alpha, \beta)$$

$$= \prod_{i=1}^{m} p(z_i|\alpha) \prod_{k=1}^{k} p(\{x_{ij}|z_{ij} = k\} \, |\beta)$$

sample z sequentially

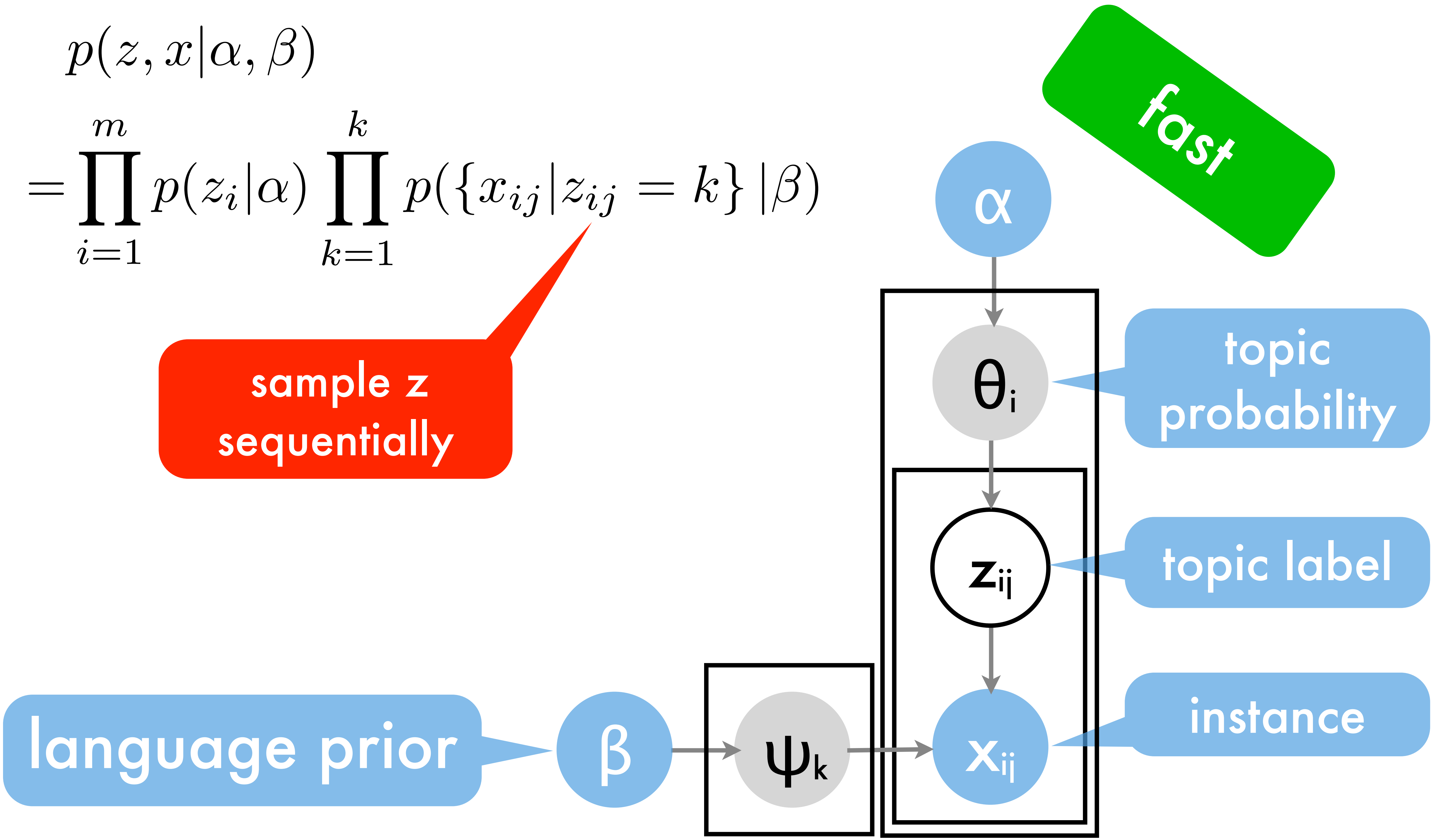


Collapsed Sampler

$$p(z, x|\alpha, \beta)$$

$$= \prod_{i=1}^{m} p(z_i|\alpha) \prod_{k=1}^{k} p(\{x_{ij}|z_{ij} = k\} |\beta)$$

sample z
sequentially



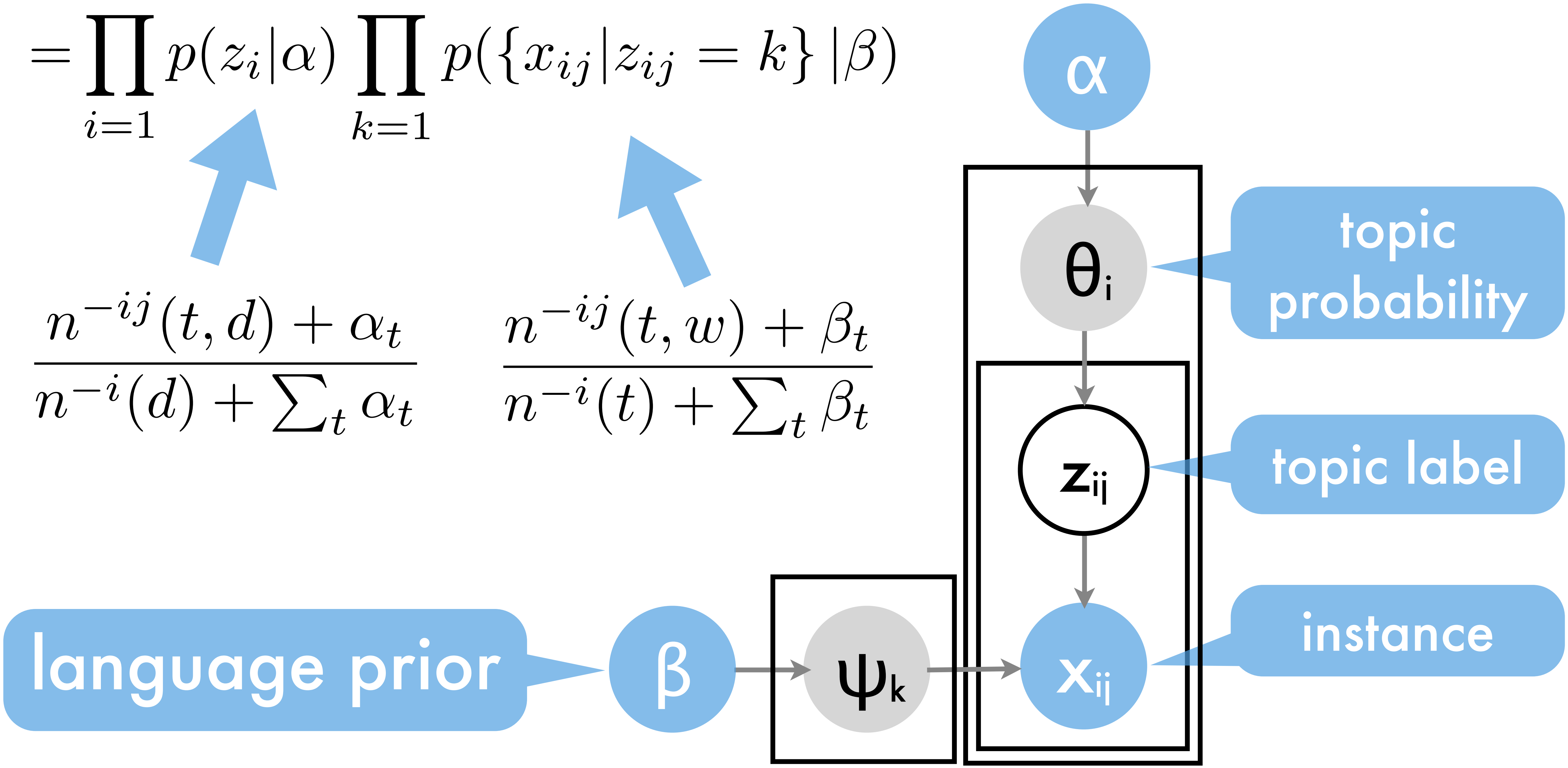
Collapsed Sampler

Griffiths & Steyvers, 2005

$$p(z, x|\alpha, \beta)$$

$$= \prod_{i=1}^{m} p(z_i|\alpha) \prod_{k=1}^{k} p(\{x_{ij}|z_{ij} = k\}\,|\beta)$$

$$\frac{n^{-ij}(t,d) + \alpha_t}{n^{-i}(d) + \sum_t \alpha_t} \qquad \frac{n^{-ij}(t,w) + \beta_t}{n^{-i}(t) + \sum_t \beta_t}$$



Collapsed Sampler

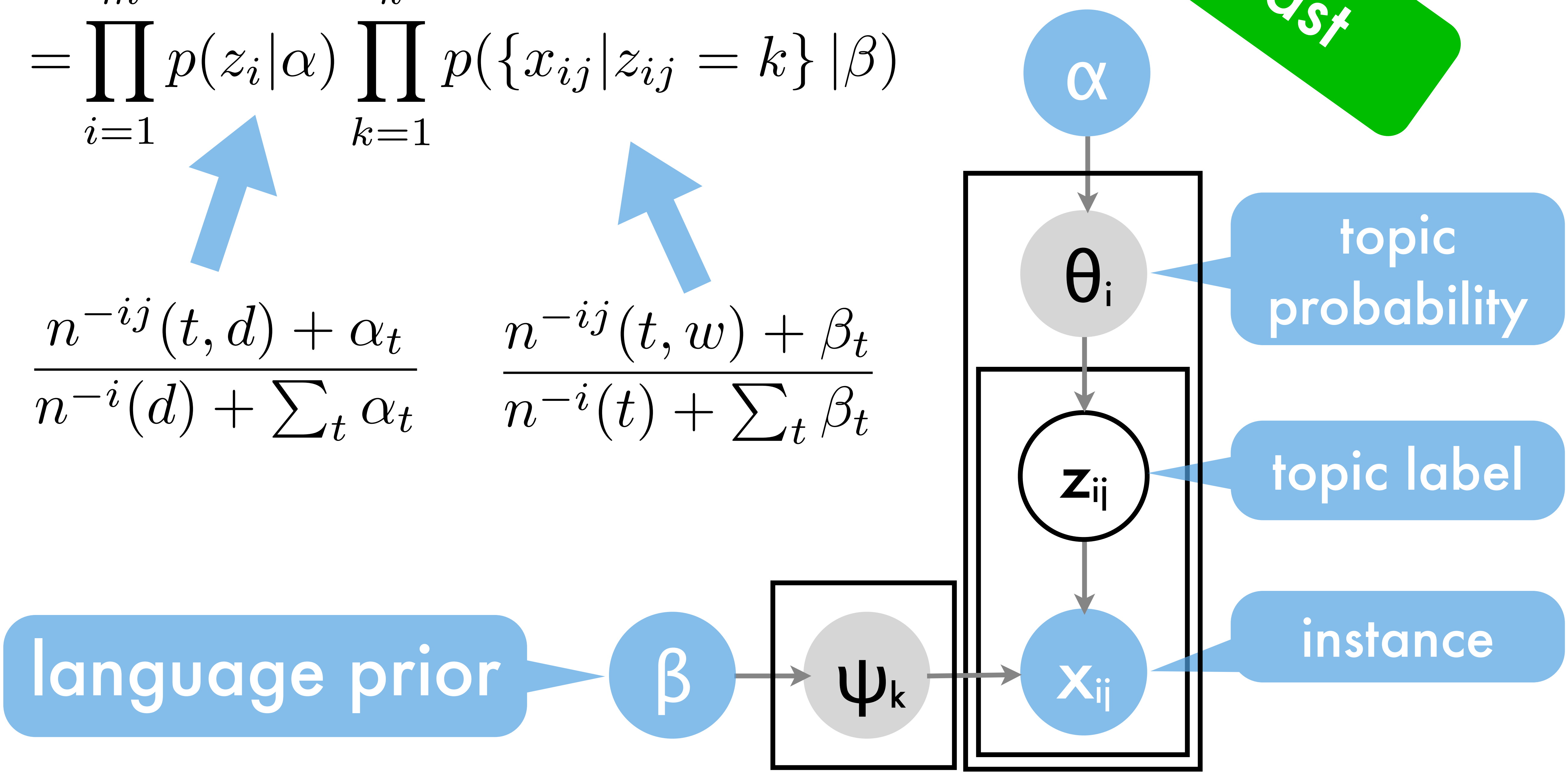
Griffiths & Steyvers, 2005

fast

$$p(z, x|\alpha, \beta)$$

$$= \prod_{i=1}^{m} p(z_i|\alpha) \prod_{k=1}^{k} p(\{x_{ij}|z_{ij} = k\} |\beta)$$

$$\frac{n^{-ij}(t,d) + \alpha_t}{n^{-i}(d) + \sum_t \alpha_t} \qquad \frac{n^{-ij}(t,w) + \beta_t}{n^{-i}(t) + \sum_t \beta_t}$$



Sequential Algorithm (Gibbs sampler)

- For 1000 iterations do
  - For each document do
    - For each word in the document do
      - Resample topic for the word
      - Update local (document, topic) table
      - Update CPU local (word, topic) table
      - Update global (word, topic) table

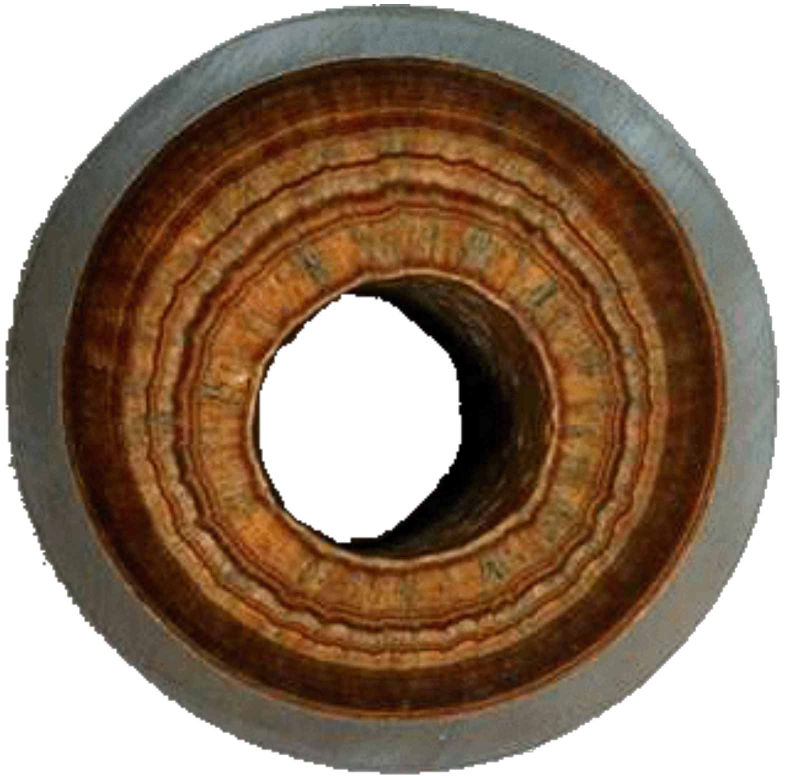
Sequential Algorithm (Gibbs sampler)

- For 1000 iterations do
  - For each document do
    - For each word in the document do
      - Resample topic for the word
      - Update local (document, topic) table
      - Update CPU local (word, topic) table
      - Update global (word, topic) table

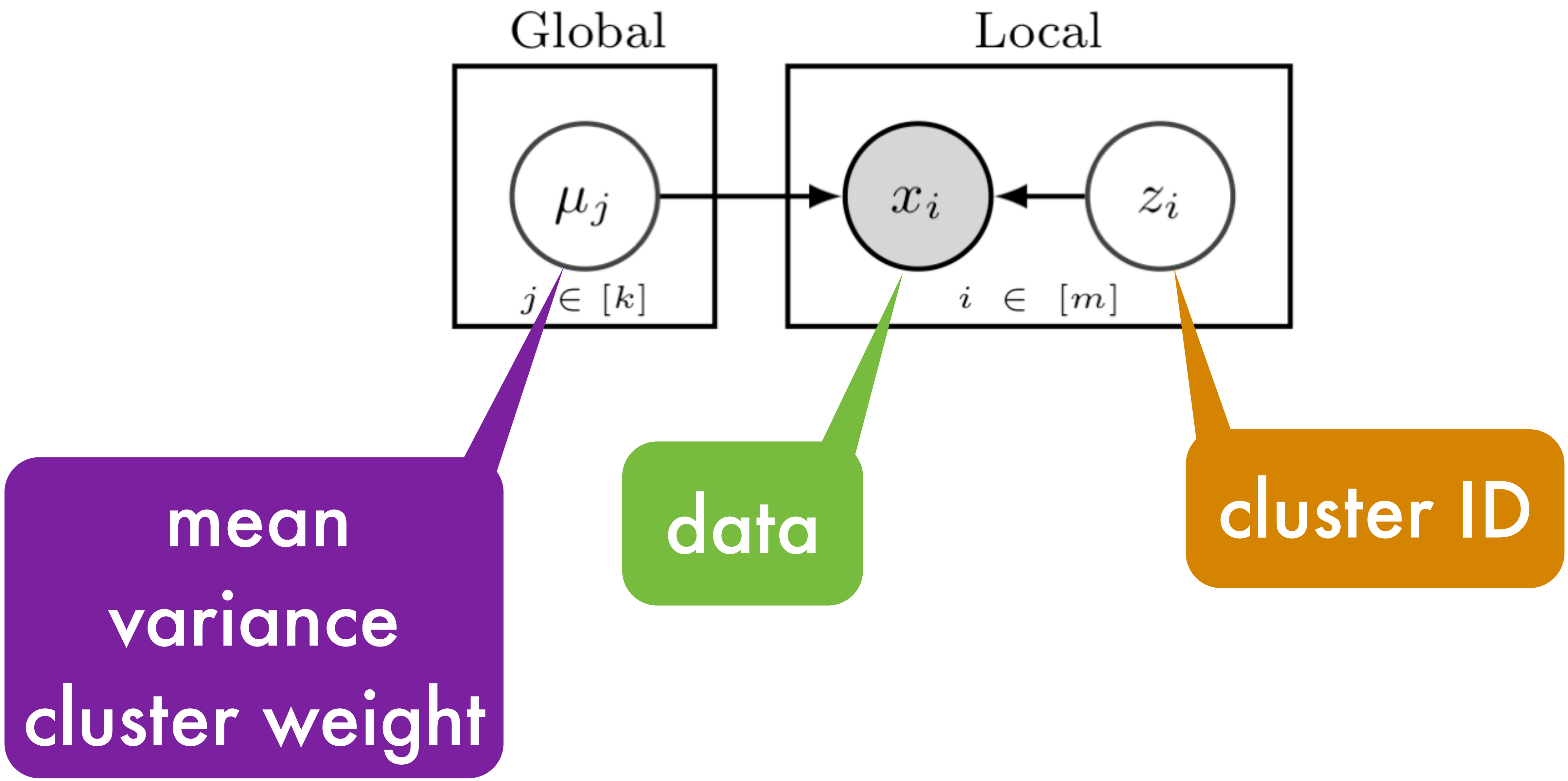
this kills parallelism

3. Design Principles

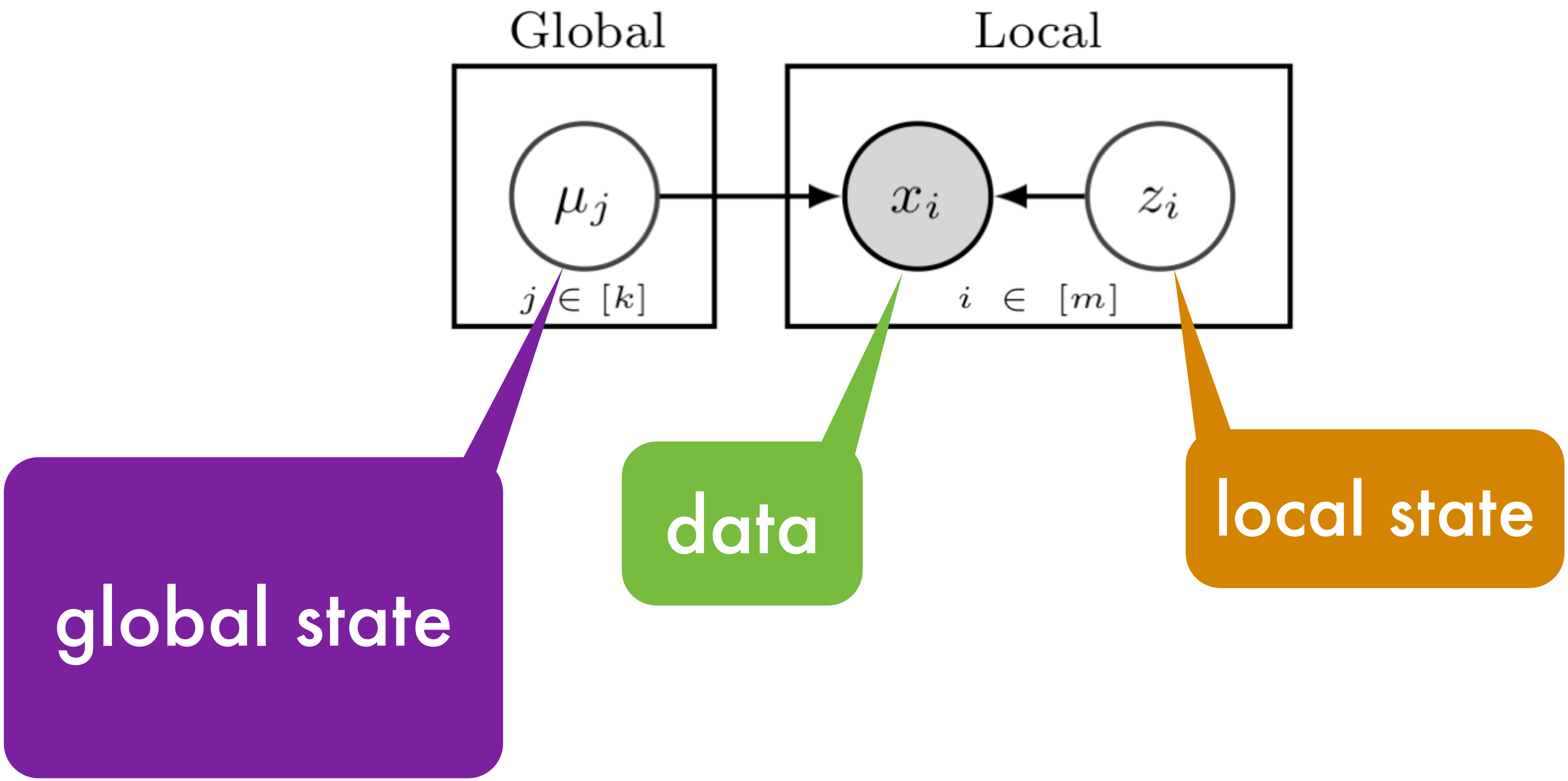
Scaling Problems



3 Problems

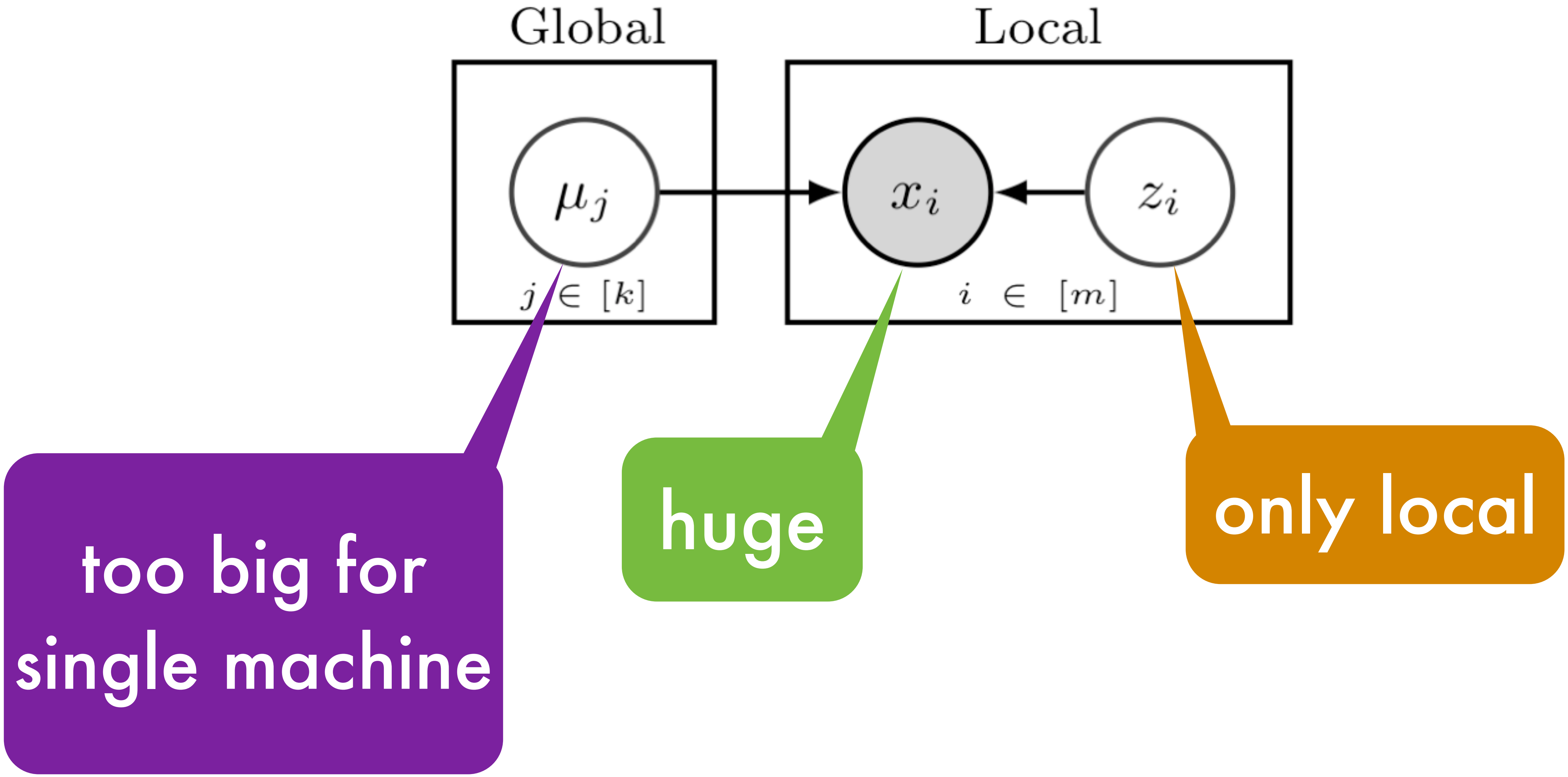

Global
Local
μ_j
x_i
z_i
$j \in [k]$
$i \in [m]$
mean
variance
cluster weight
data
cluster ID

3 Problems



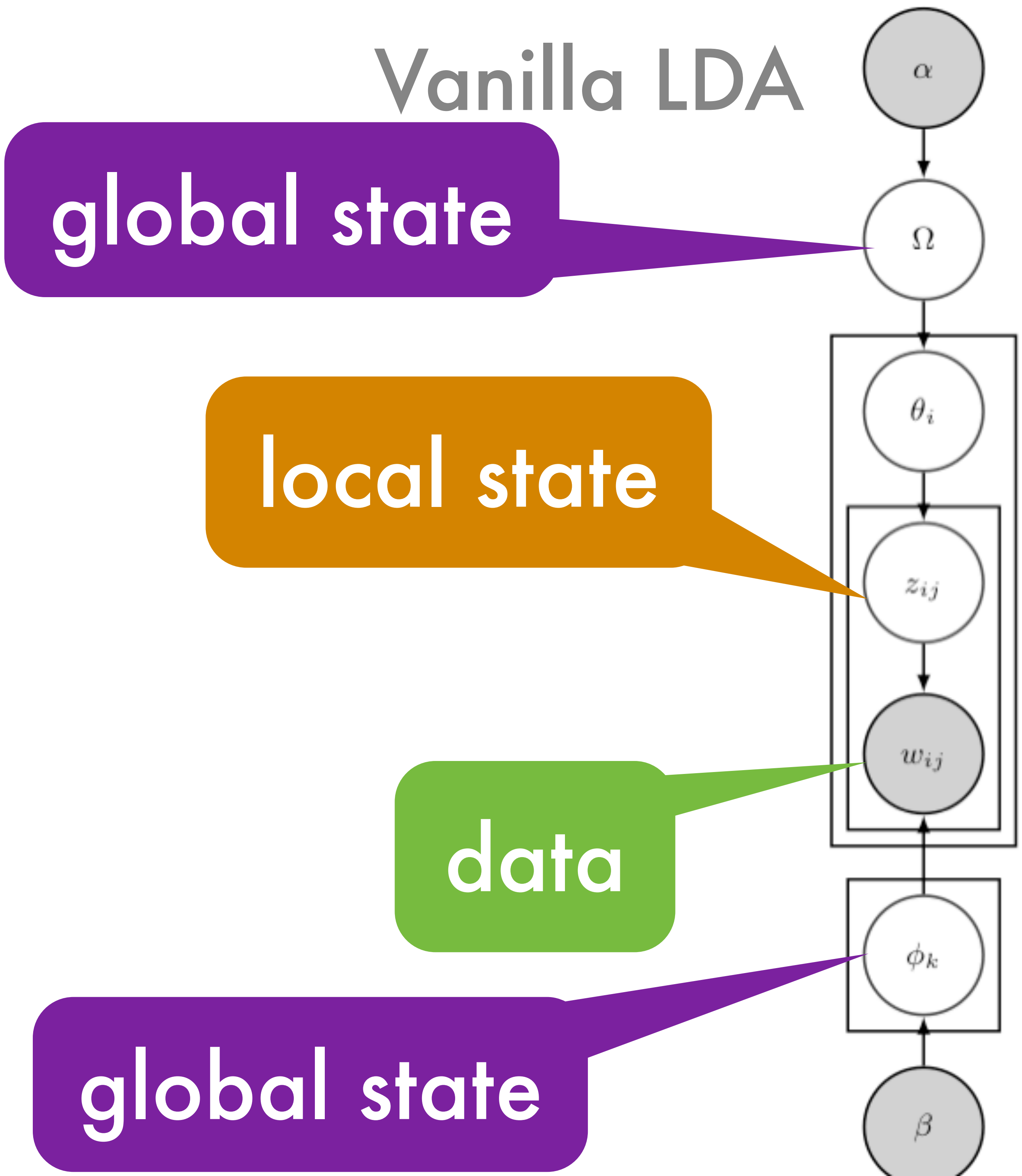
YAHOO!

3 Problems

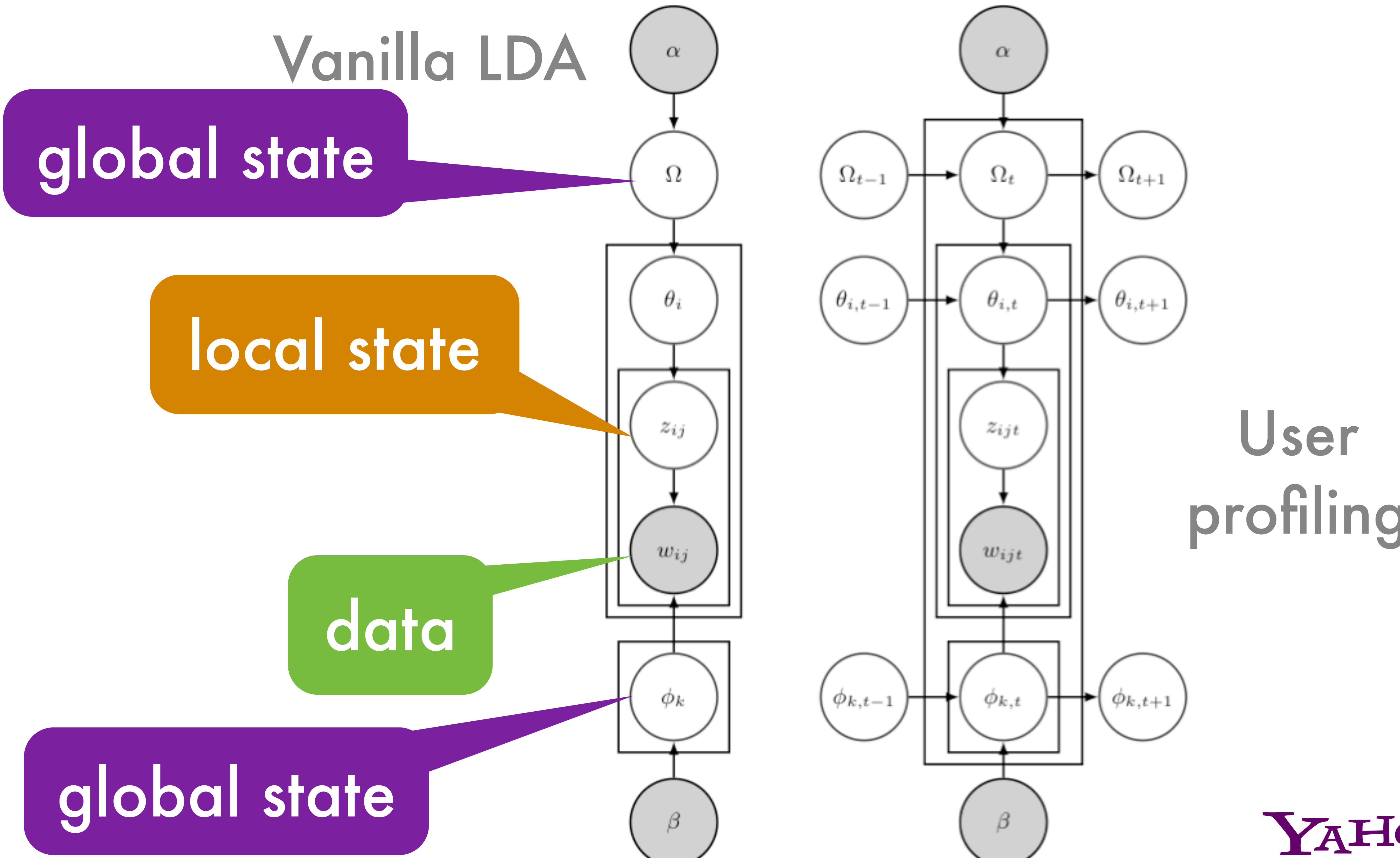

Global
Local
μ_j
x_i
z_i
$j \in [k]$
$i \in [m]$
too big for single machine
huge
only local

3 Problems

Vanilla LDA



3 Problems

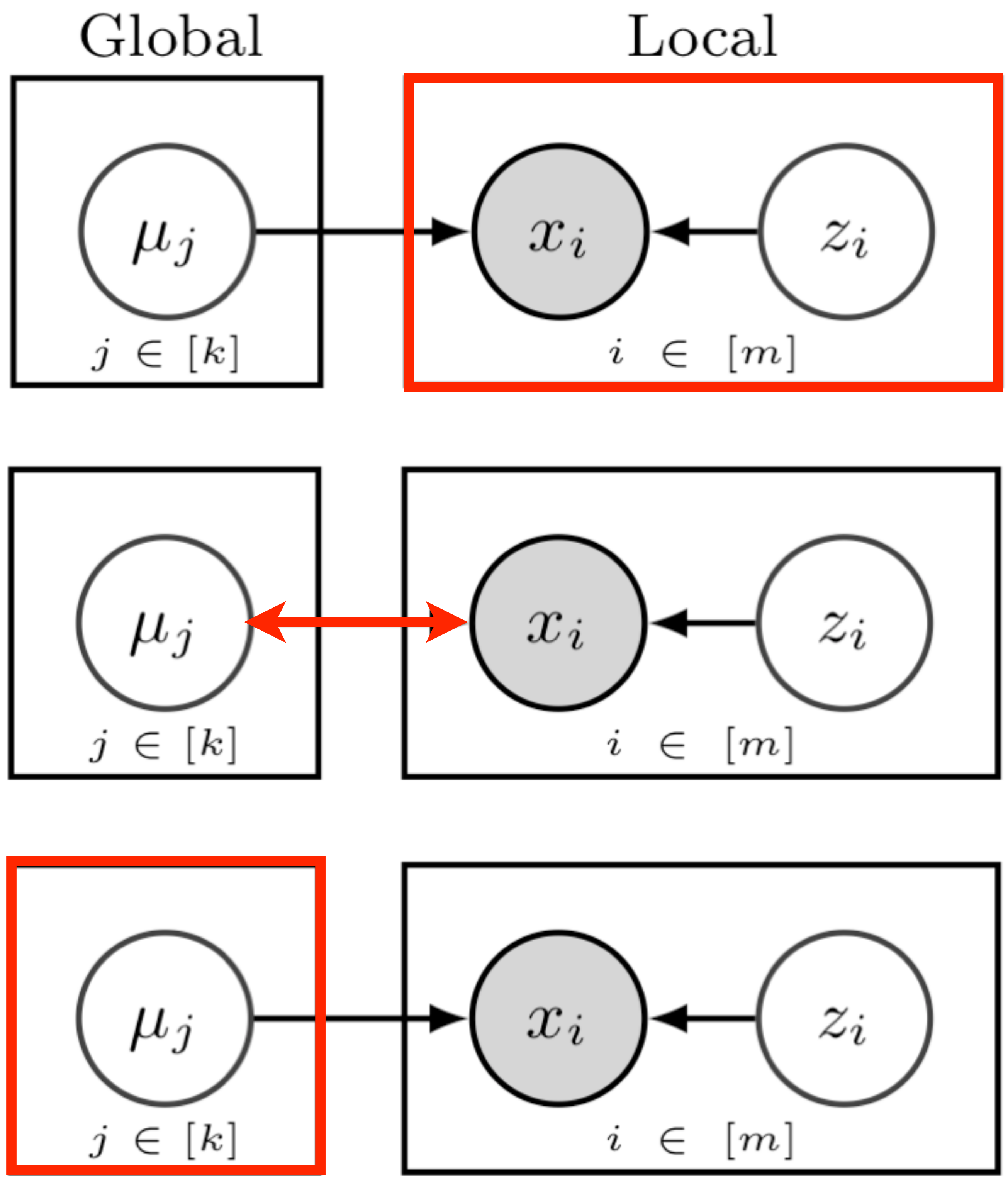


Vanilla LDA
global state
local state
data
global state
User profiling
α
Ω
θ_i
z_{ij}
w_{ij}
ϕ_k
β
Ω_{t-1}
Ω_t
Ω_{t+1}
$\theta_{i,t-1}$
$\theta_{i,t}$
$\theta_{i,t+1}$
z_{ijt}
w_{ijt}
$\phi_{k,t-1}$
$\phi_{k,t}$
$\phi_{k,t+1}$

3 Problems

local state
is too large

global state
is too large



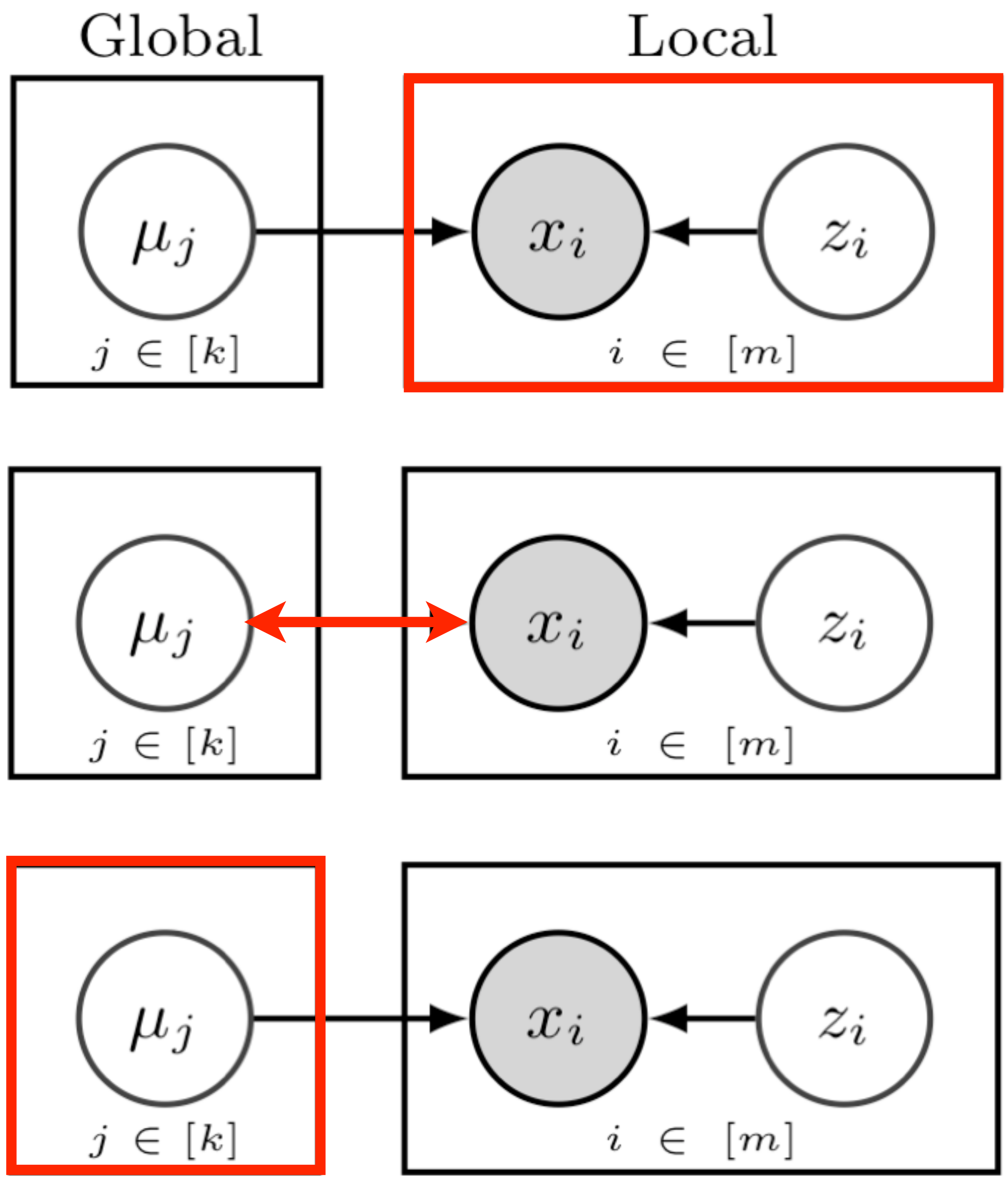
does not fit
into memory

network load
& barriers

does not fit
into memory

3 Problems

local state
is too large



stream local
data from disk

network load
& barriers

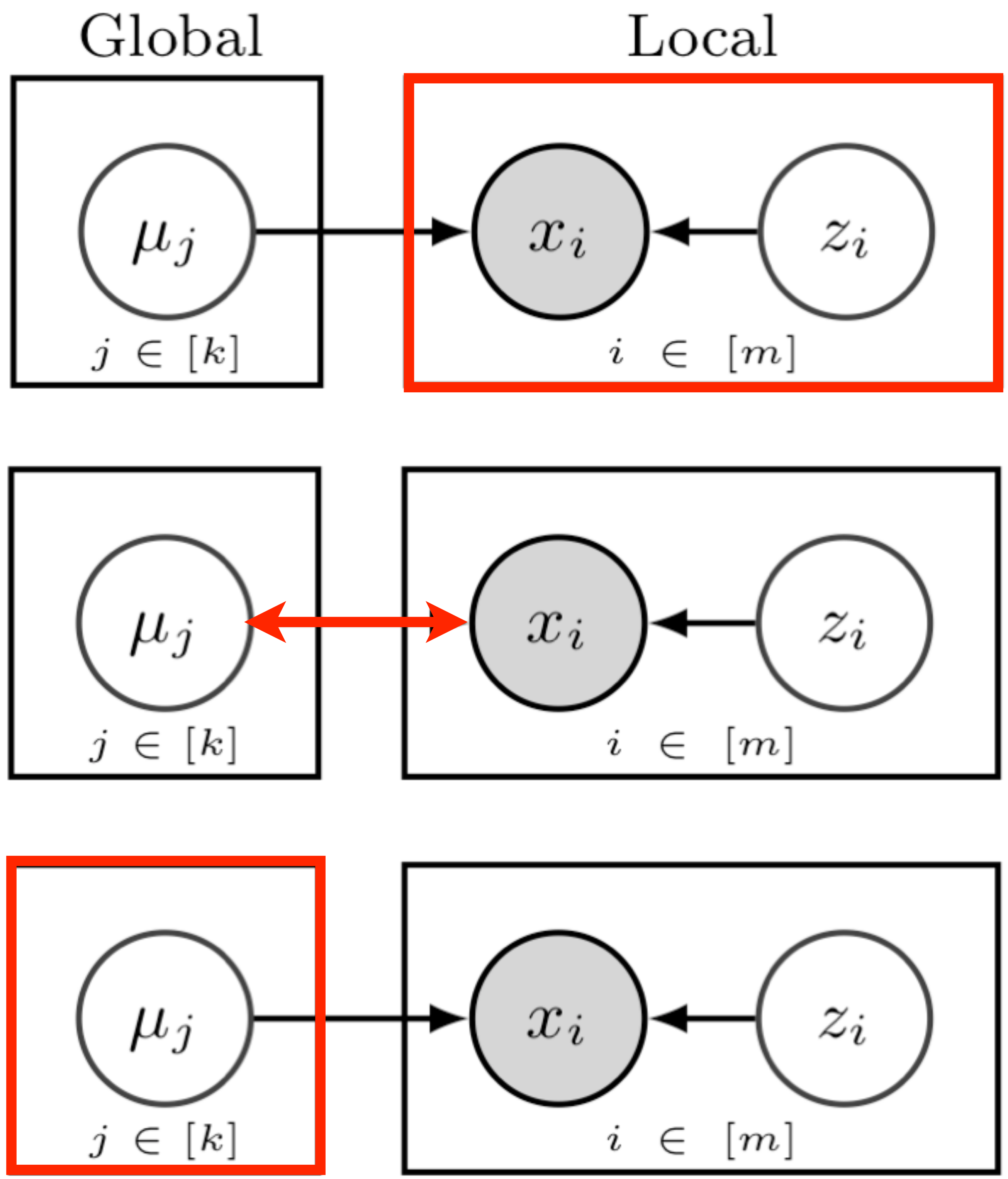
global state
is too large

does not fit
into memory

3 Problems

local state is too large

global state is too large



stream local data from disk

asynchronous synchronization

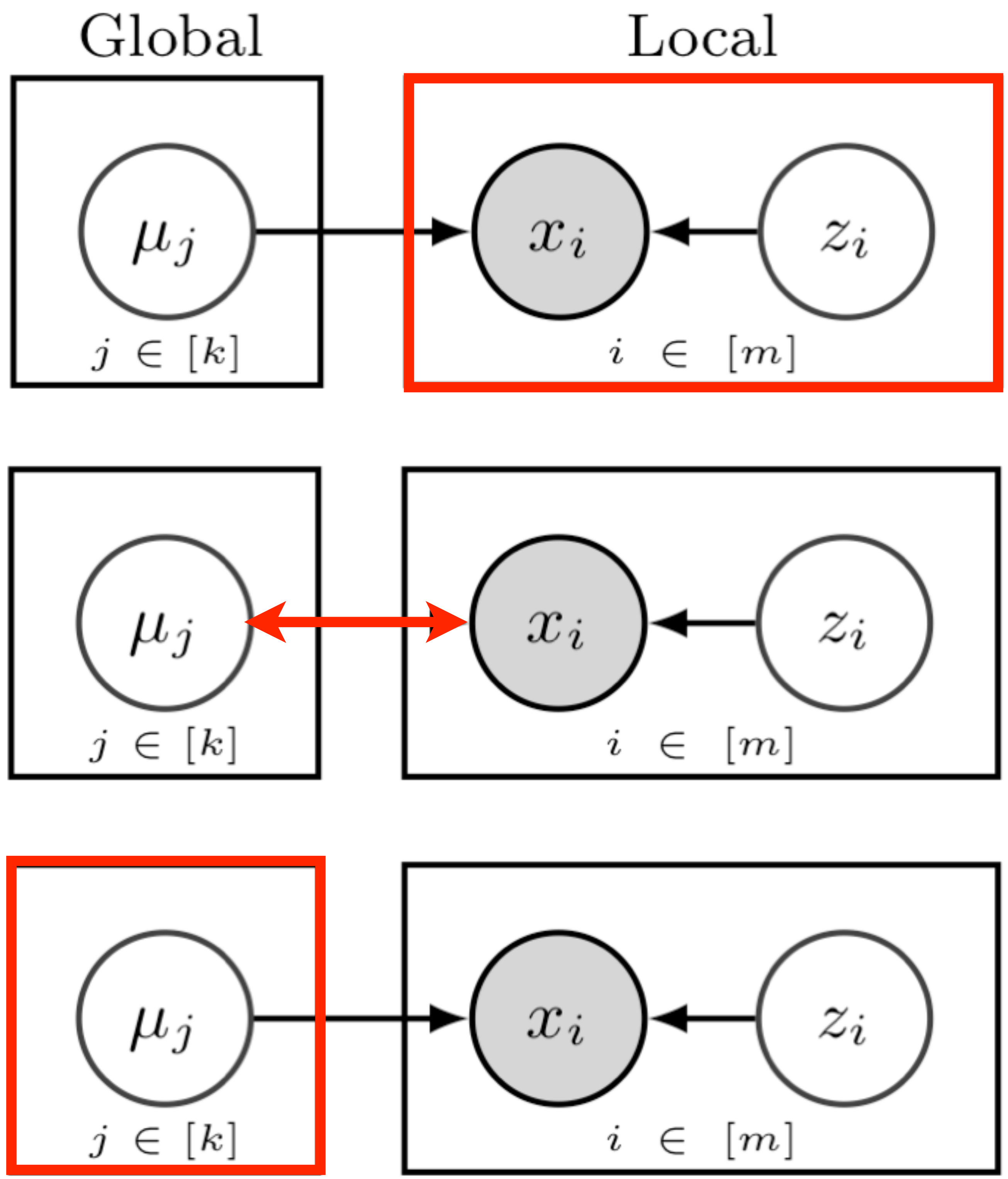
does not fit into memory

YAHOO!

3 Problems

local state
is too large

global state
is too large



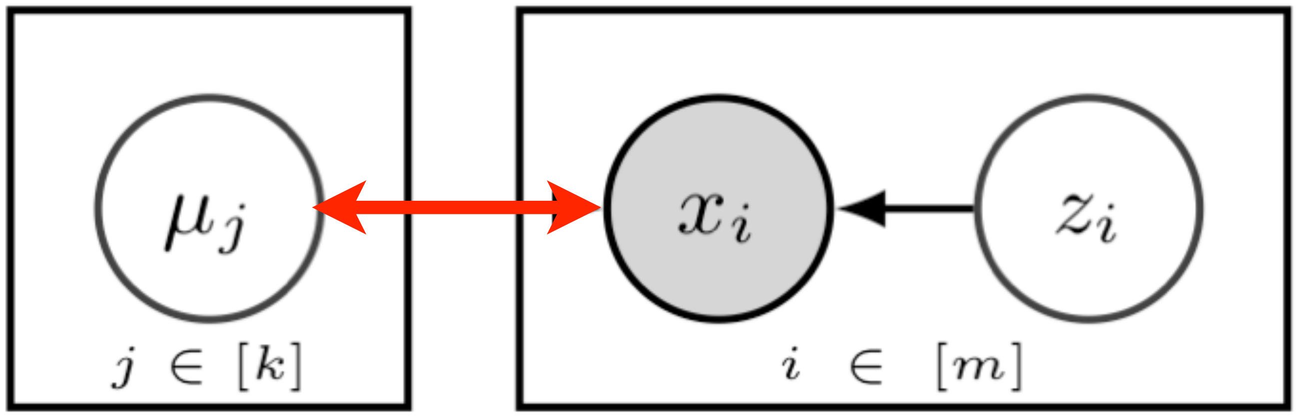
stream local data from disk

asynchronous synchronization

partial view

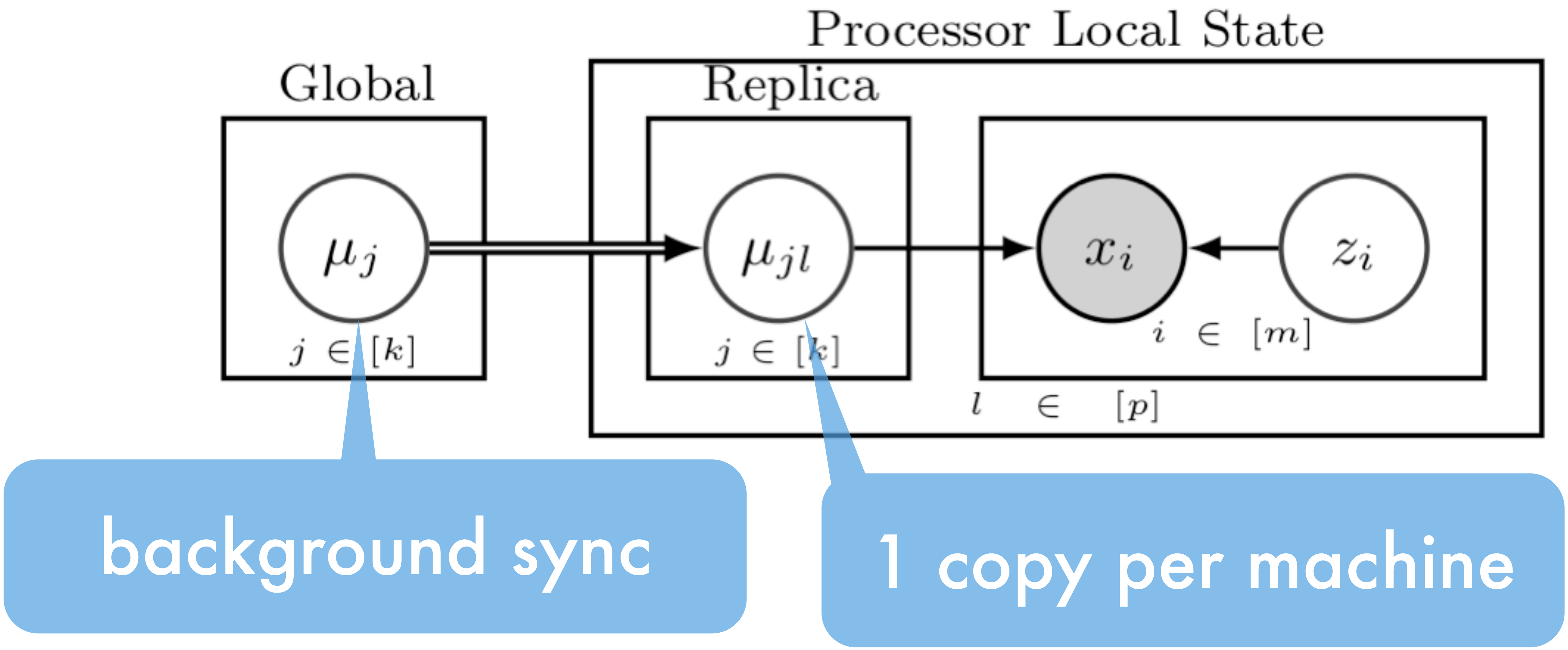
Global state synchronization

Lock and Barrier



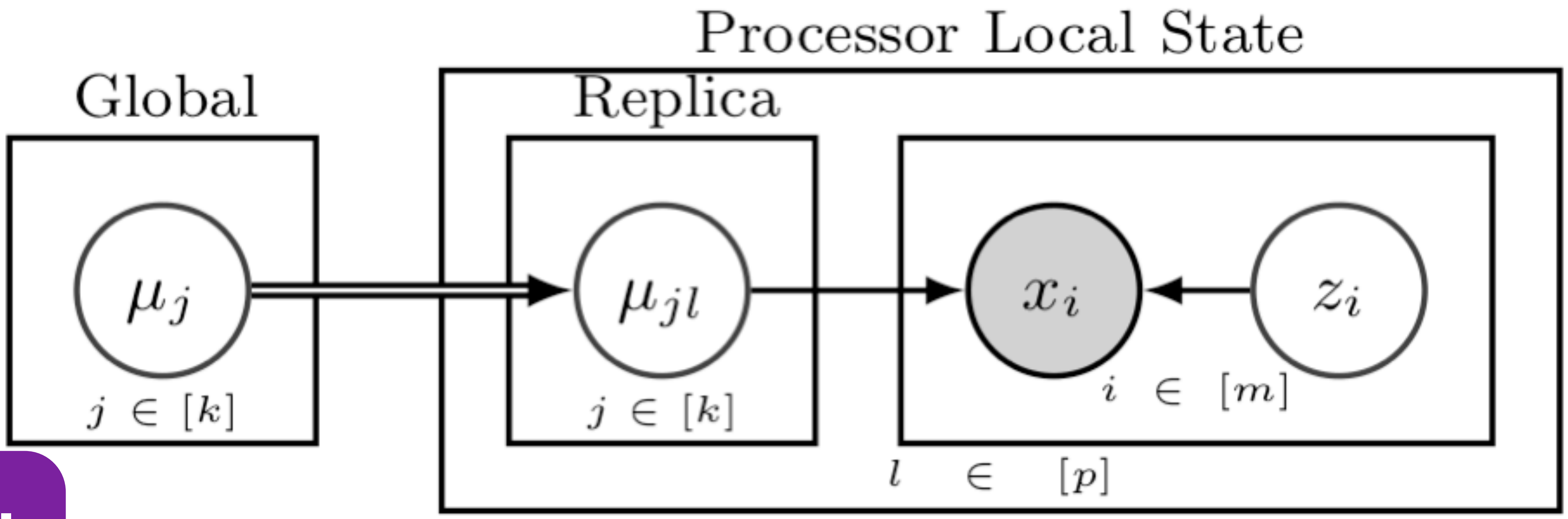
- Changes in μ affect distribution in z
- Changes in z affect distribution in μ
(in particular for a collapsed sampler)
- Lock z, then recompute μ
- Lock all but single z_i (for collapsed sampler)

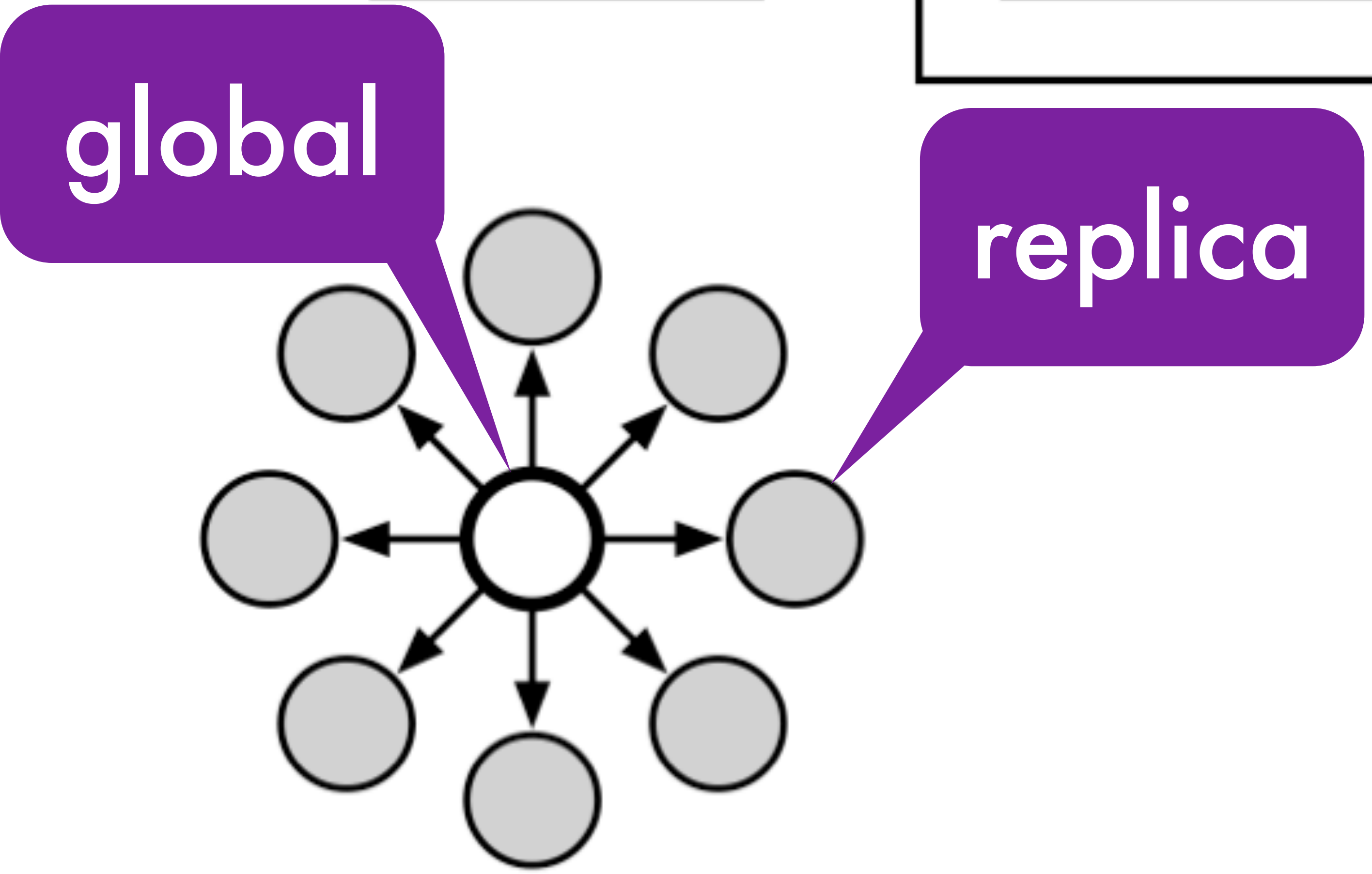
Variable replication



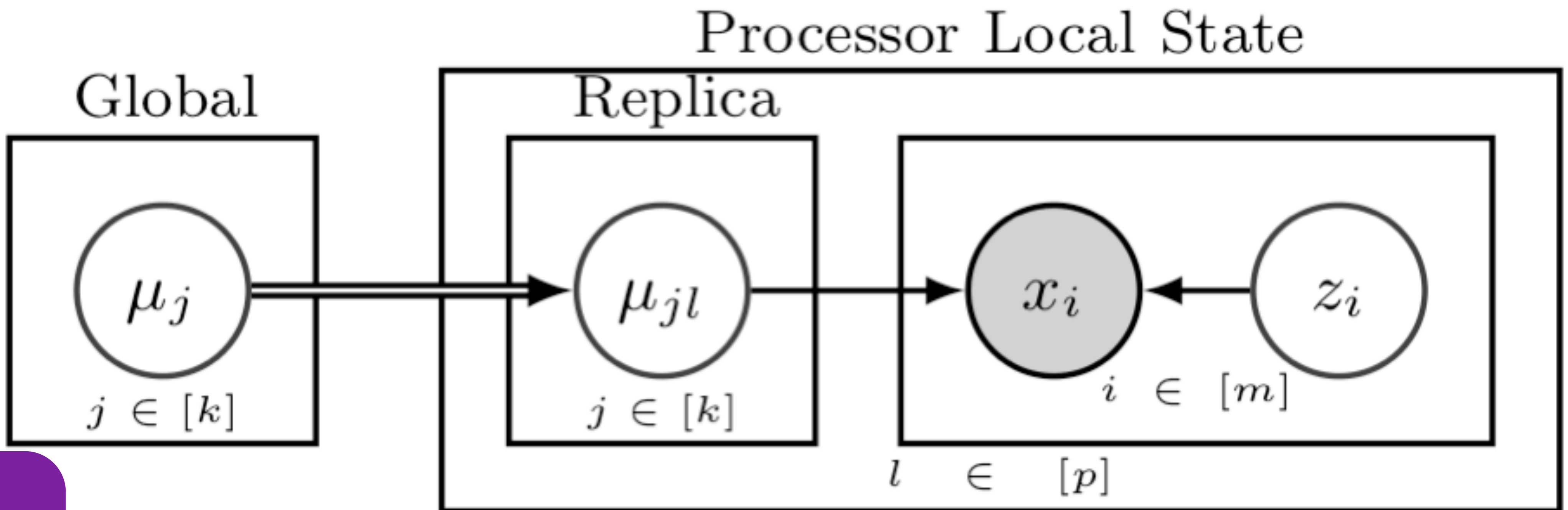
- No locks between machines to access z
- Synchronization mechanism for global μ needed
- In LDA this is the local copy of the (topic,word) counts

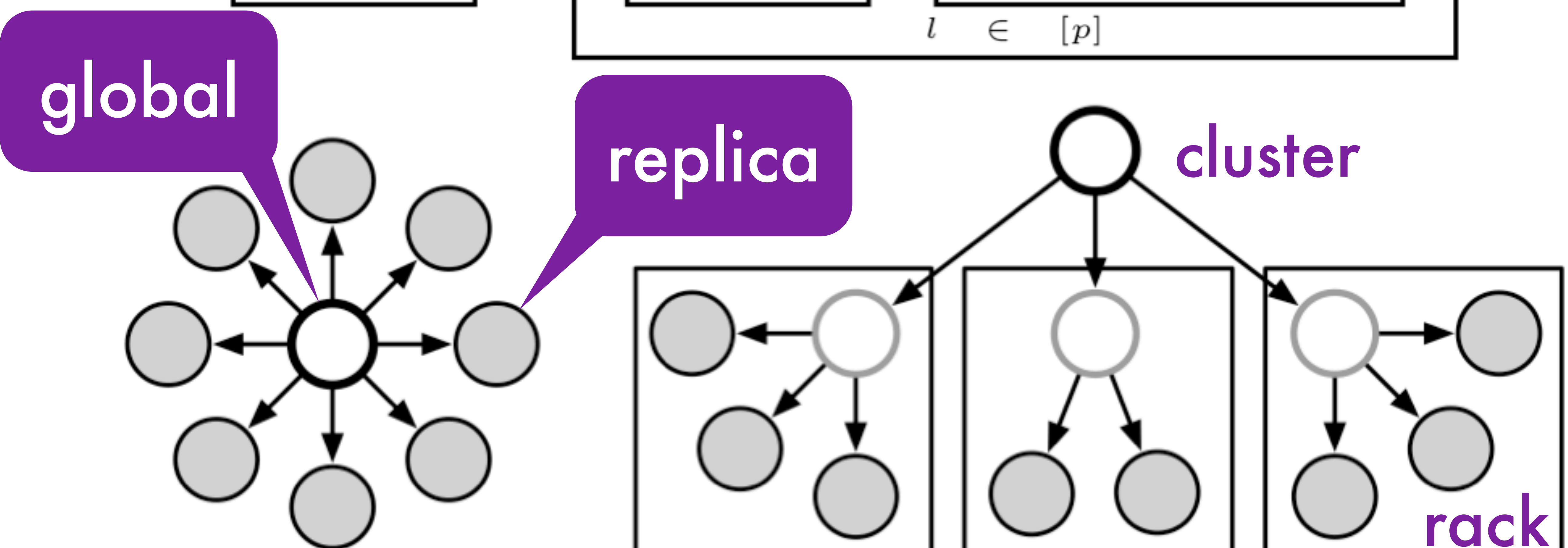
Distribution





Distribution



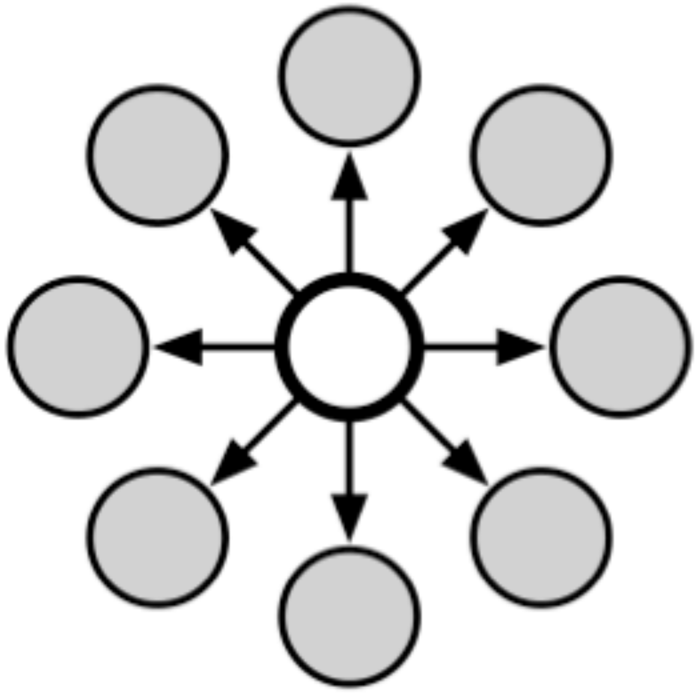


Message Passing

- Start with common state
- Child stores old and new state
- Parent keeps global state
- Transmit differences asynchronously
  - Inverse element for difference
  - Abelian group for commutativity
    (sum, log-sum, cyclic group, exponential families)

local to global

$$\delta \leftarrow x - x^{\text{old}}$$
$$x^{\text{old}} \leftarrow x$$
$$x^{\text{global}} \leftarrow x^{\text{global}} + \delta$$



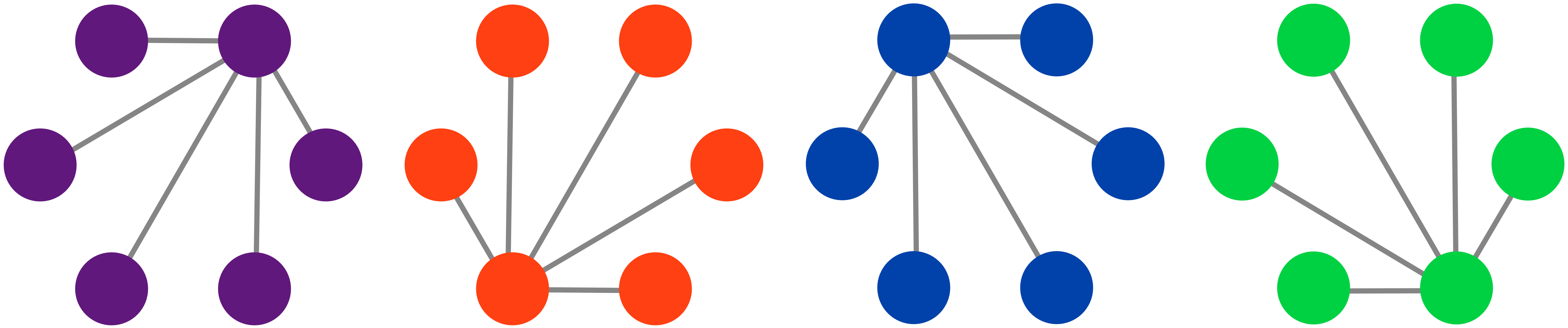
global to local

$$x \leftarrow x + (x^{\text{global}} - x^{\text{old}})$$
$$x^{\text{old}} \leftarrow x^{\text{global}}$$

Distribution

- Dedicated server for variables
  - Insufficient bandwidth (hotspots)
  - Insufficient memory
- Select server via consistent hashing
  (random trees a la Karger et al. 1999 if needed)

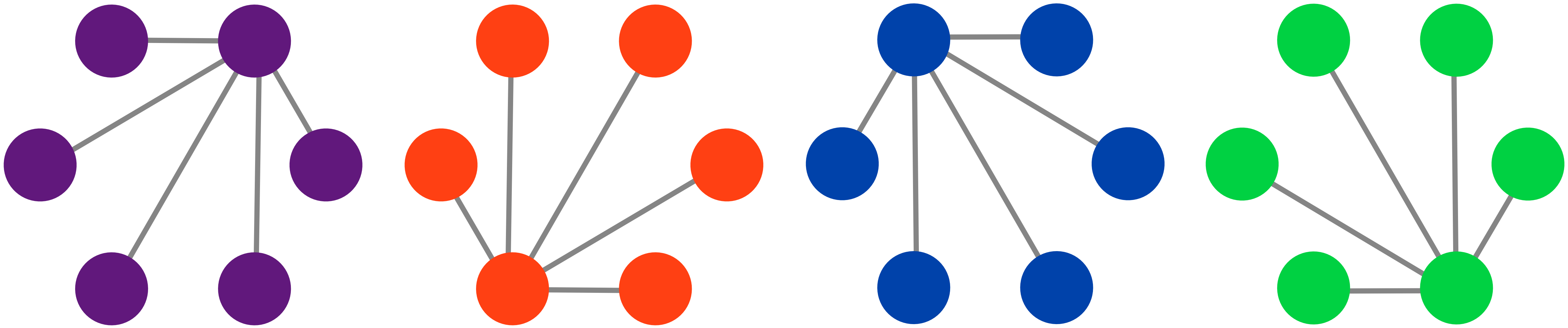
$$m(x) = \operatorname*{argmin}_{m \in M} h(x, m)$$



Distribution & fault tolerance

- Storage is O(1/k) per machine
- Communication is O(1) per machine
- Fast snapshots O(1/k) per machine (stop sync and dump state per vertex)

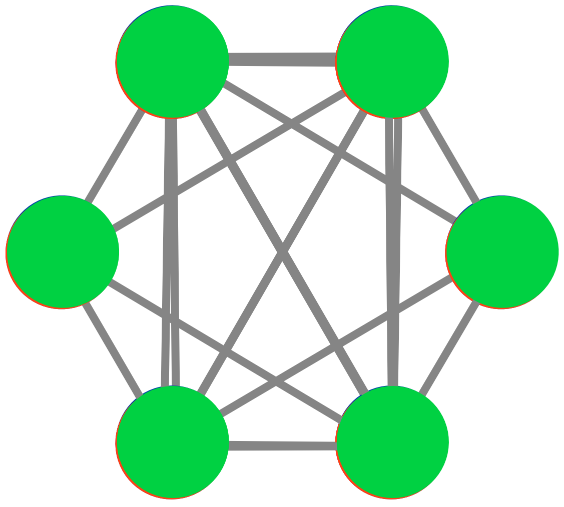
$$m(x) = \underset{m \in M}{\operatorname{argmin}}\, h(x, m)$$



Distribution & fault tolerance

- Storage is O(1/k) per machine
- Communication is O(1) per machine
- Fast snapshots O(1/k) per machine (stop sync and dump state per vertex)

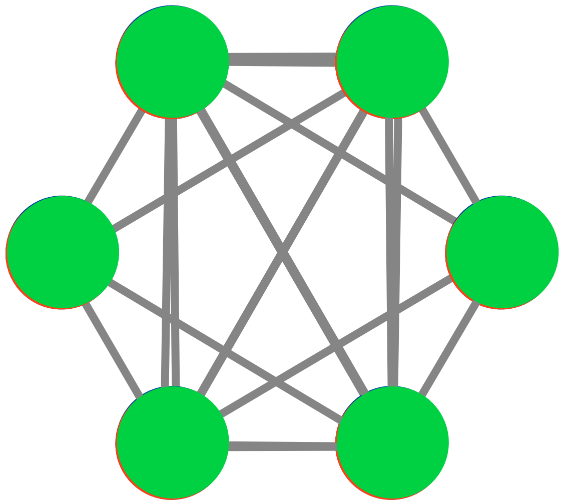
$$m(x) = \operatorname*{argmin}_{m \in M} h(x, m)$$



Distribution & fault tolerance

- Storage is O(1/k) per machine
- Communication is O(1) per machine
- Fast snapshots O(1/k) per machine (stop sync and dump state per vertex)
- O(k) open connections per machine
- O(1/k) throughput per machine

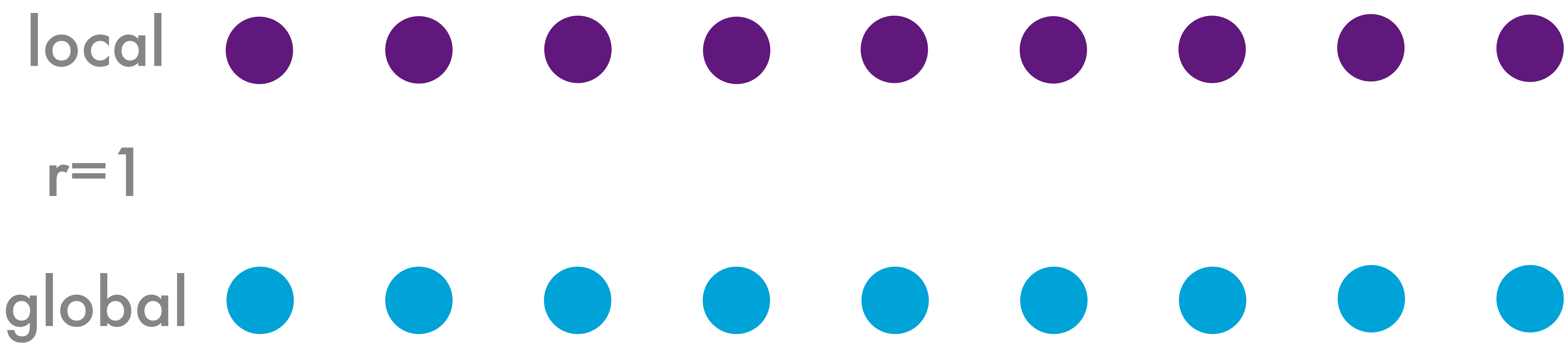
$$m(x) = \operatorname*{argmin}_{m \in M} h(x, m)$$



Synchronization Protocol

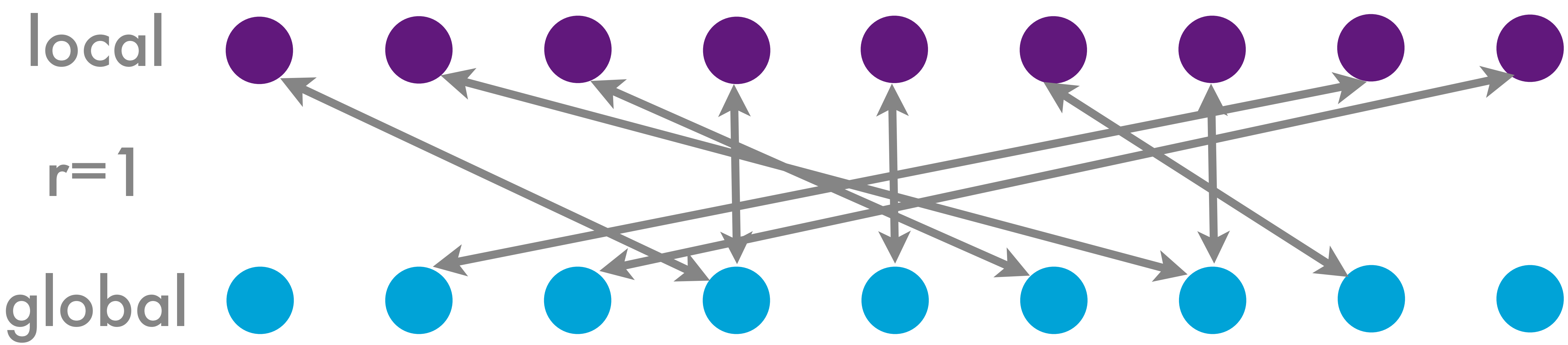
Synchronization

- Data rate between machines is O(1/k)
- Machines operate asynchronously (barrier free)
- Solution
  - Schedule message pairs
  - Communicate with r random machines simultaneously



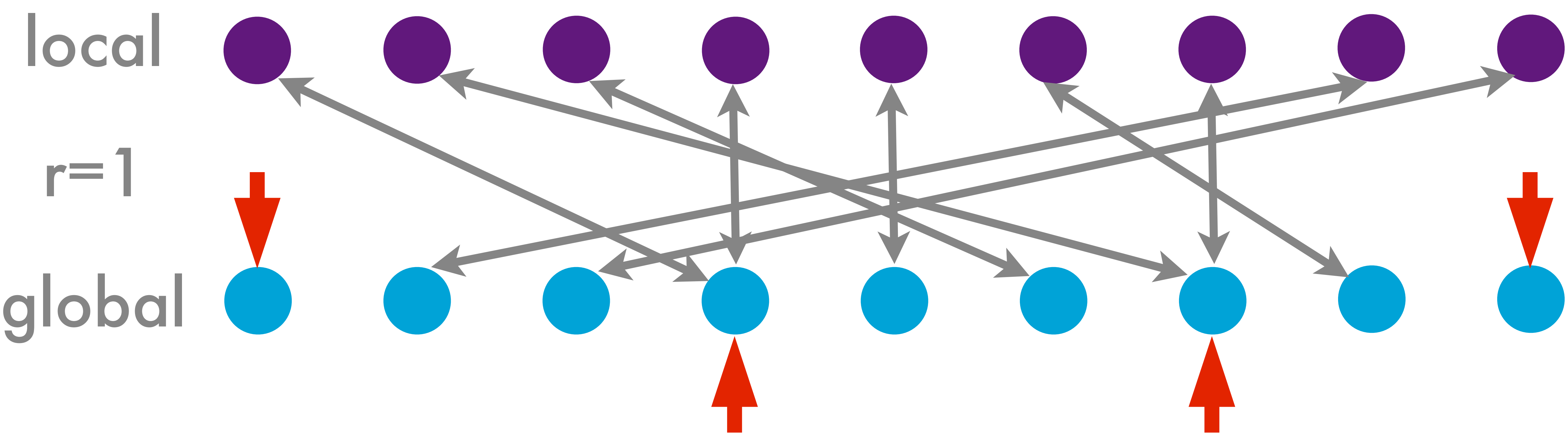
Synchronization

- Data rate between machines is O(1/k)
- Machines operate asynchronously (barrier free)
- Solution
  - Schedule message pairs
  - Communicate with r random machines simultaneously



Synchronization

- Data rate between machines is O(1/k)
- Machines operate asynchronously (barrier free)
- Solution
  - Schedule message pairs
  - Communicate with r random machines simultaneously



Synchronization

- Data rate between machines is O(1/k)
- Machines operate asynchronously (barrier free)
- Solution
  - Schedule message pairs
  - Communicate with r random machines simultaneously
  - Use Luby-Rackoff PRPG for load balancing
- Efficiency guarantee

$$1 - e^{-r} \sum_{i=0}^{r} \left[1 - \frac{i}{r}\right] \frac{r^i}{i!} \leq \text{Eff} \leq 1 - e^{-r}$$

4 simultaneous connections are sufficient

Architecture

Sequential Algorithm (Gibbs sampler)

- For 1000 iterations do
  - For each document do
    - For each word in the document do
      - Resample topic for the word
      - Update local (document, topic) table
      - Update CPU local (word, topic) table
      - Update global (word, topic) table

Sequential Algorithm (Gibbs sampler)

- For 1000 iterations do
  - For each document do
    - For each word in the document do
      - Resample topic for the word
      - Update local (document, topic) table
      - Update CPU local (word, topic) table
      - Update global (word, topic) table

this kills parallelism

Distributed asynchronous sampler

- For 1000 iterations do (independently per computer)
  - For each thread/core do
    - For each document do
      - For each word in the document do
        - Resample topic for the word
        - Update local (document, topic) table
        - Generate computer local (word, topic) message
    - In parallel update local (word, topic) table
  - In parallel update global (word, topic) table

Distributed asynchronous sampler

- For 1000 iterations do (independently per computer)
  - For each thread/core do
    - For each document do
      - For each word in the document do
        - Resample topic for the word
        - Update local (document, topic) table
        - Generate computer local (word, topic) message
    - In parallel update local (word, topic) table
  - In parallel update global (word, topic) table

concurrent cpu hdd net

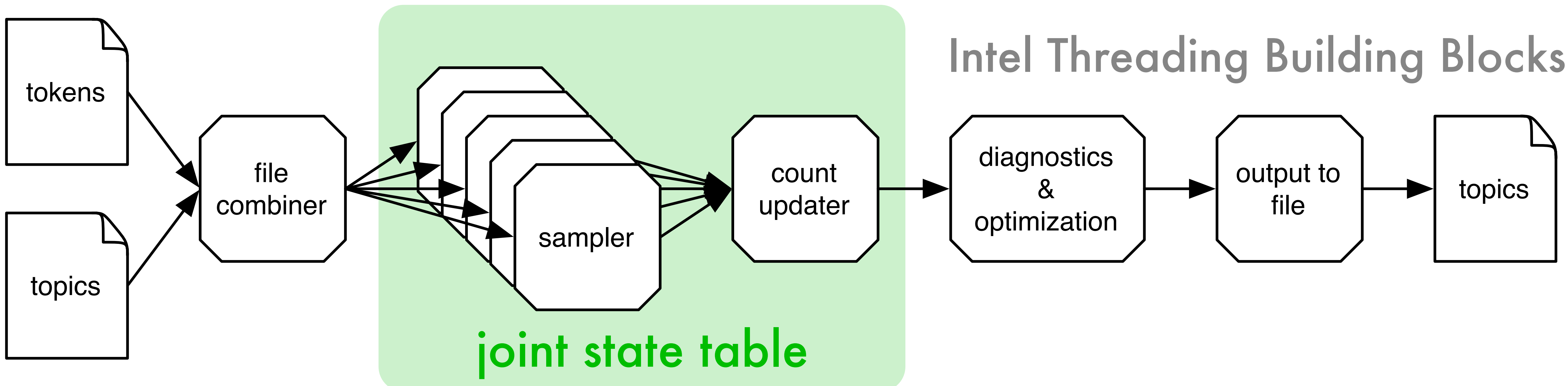
minimal view

continuous sync

barrier free

YAHOO!

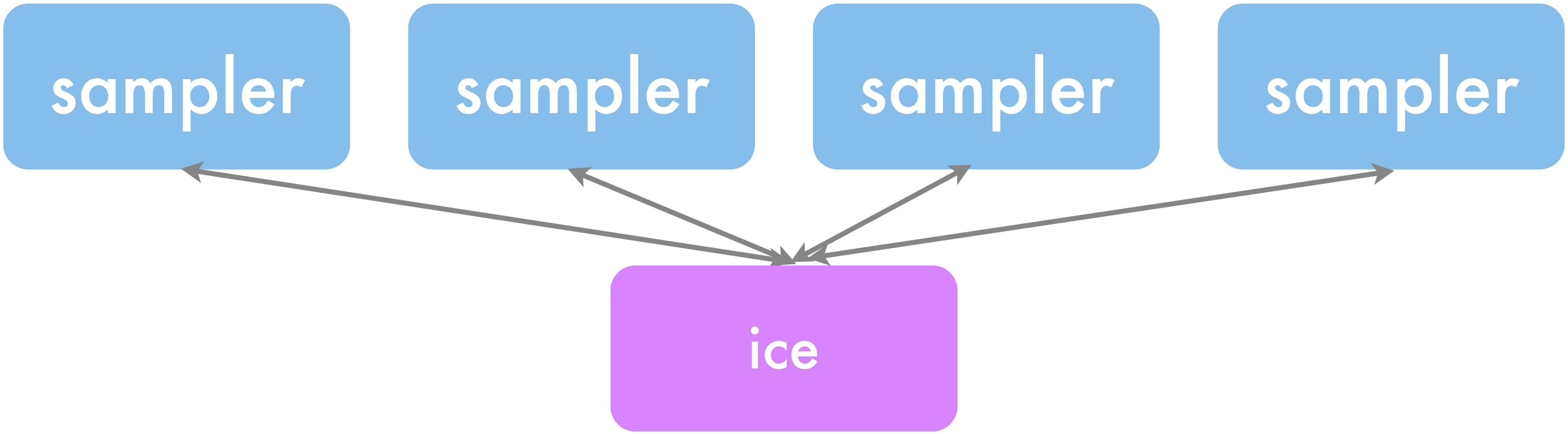
Multicore Architecture



- Decouple multithreaded sampling and updating (almost) avoids stalling for locks in the sampler
- Joint state table
  - much less memory required
  - samplers syncronized (10 docs vs. millions delay)
- Hyperparameter update via stochastic gradient descent
- No need to keep documents in memory (streaming)

YAHOO!

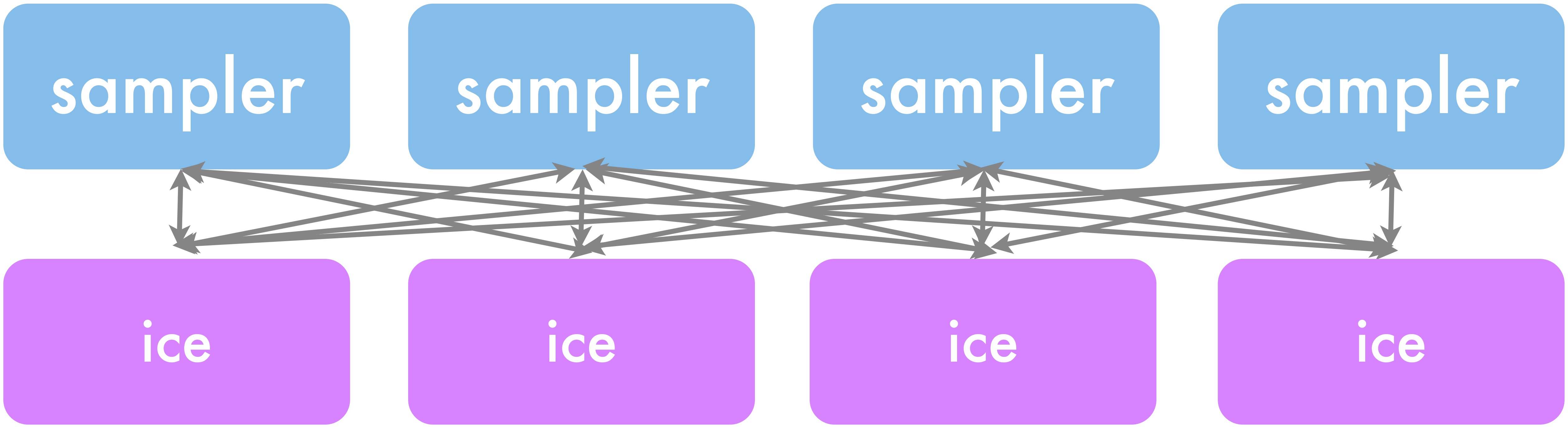
Cluster Architecture



- Distributed (key,value) storage via ICE
- Background asynchronous synchronization
  - single word at a time to avoid deadlocks
  - no need to have joint dictionary
  - uses disk, network, cpu simultaneously

YAHOO!

Cluster Architecture



- Distributed (key,value) storage via ICE
- Background asynchronous synchronization
  - single word at a time to avoid deadlocks
  - no need to have joint dictionary
  - uses disk, network, cpu simultaneously

YAHOO!

Making it work

- Startup
  - Naive: randomly initialize topics on each node (read from disk if already assigned - hotstart)
  - Forward sampling for startup much faster
  - Aggregate changes on the fly
- Failover
  - State constantly being written to disk (worst case we lose 1 iteration out of 1000)
  - Restart via standard startup routine
- Achilles heel: need to restart from checkpoint if even a single machine dies.

Easily extensible

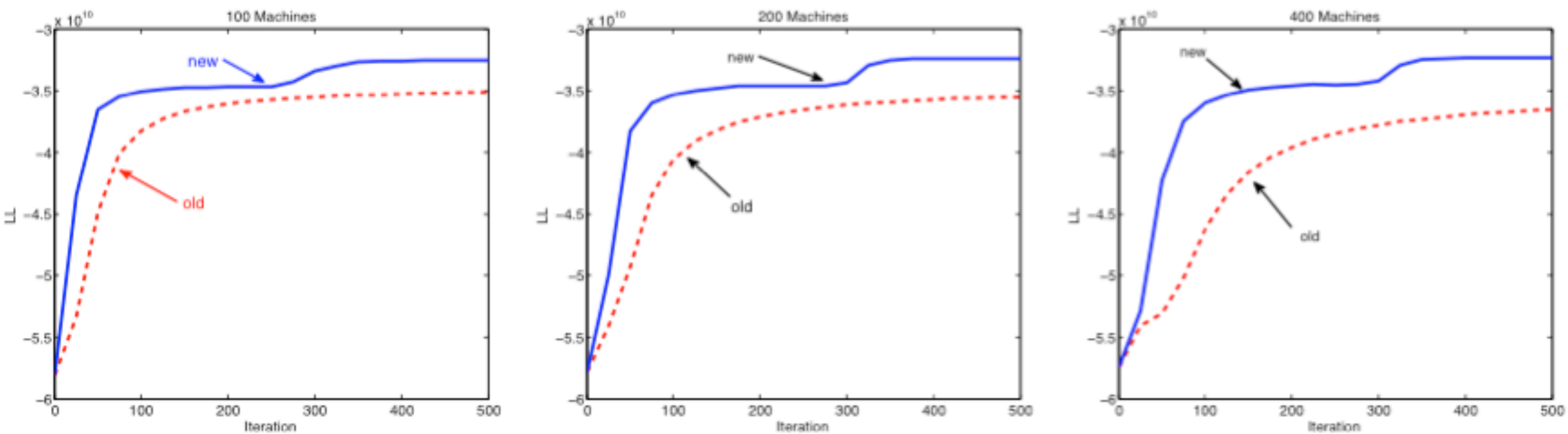
- Better language model (topical n-grams)
  can process millions of users (vs 1000s)
- Conditioning on side information (upstream)
  estimate topic based on authorship, source, joint user model ...
- Conditioning on dictionaries (downstream)
  integrate topics between different languages
- Time dependent sampler for user model
  approximate inference per episode

	Google LDA	Mallet	Irvine'08	Irvine'09	Yahoo LDA
Multicore	no	yes	yes	yes	yes
Cluster	MPI	no	MPI	point 2 point	ICE
State table	dictionary split	separate sparse	separate	separate	joint sparse
Schedule	synchronous exact	synchronous exact	synchronous exact	asynchronous approximate messages	asynchronous exact

Speed (2010 numbers)

- **1M documents per day** on 1 computer
(1000 topics per doc, 1000 words per doc)
- **350k documents per day** per node
(context switches & memcached & stray reducers)
- 8 Million docs (Pubmed)
(sampler does not burn in well - too short doc)
  - Irvine: 128 machines, 10 hours
  - Yahoo: 1 machine, 11 days
  - Yahoo: 20 machines, 9 hours
- 20 Million docs (Yahoo! News Articles)
  - Yahoo: 100 machines, 12 hours

Fast sampler



- 8 Million documents, 1000 topics, {100,200,400} machines, LDA
- Red (symmetric latency bound message passing)
- Blue (asynchronous bandwidth bound message passing & message scheduling)
  - 10x faster synchronization time
  - 10x faster snapshots
  - Scheduling improves 10% already on 150 machines

Roadmap

- Tools
  - Load distribution, balancing and synchronization
  - Clustering, Topic Models
- Models
  - Dynamic non-parametric models
  - Sequential latent variable models
- Inference Algorithms
  - Distributed batch
  - Sequential Monte Carlo
- Applications
  - User profiling
  - News content analysis & recommendation