Techniques for Dimensionality Reduction

Topic Models/LDA

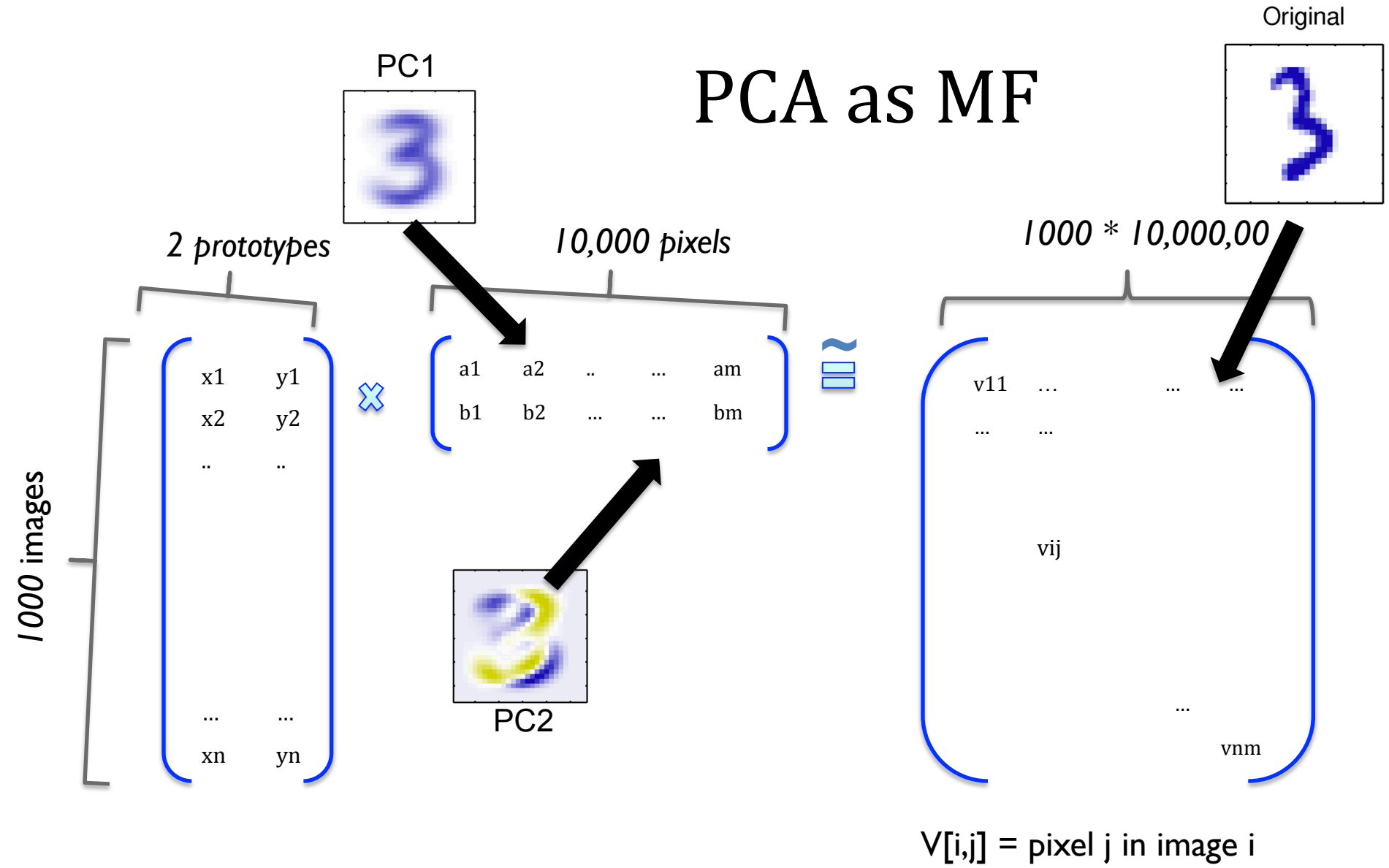
Latent Dirichlet Allocation

Outline

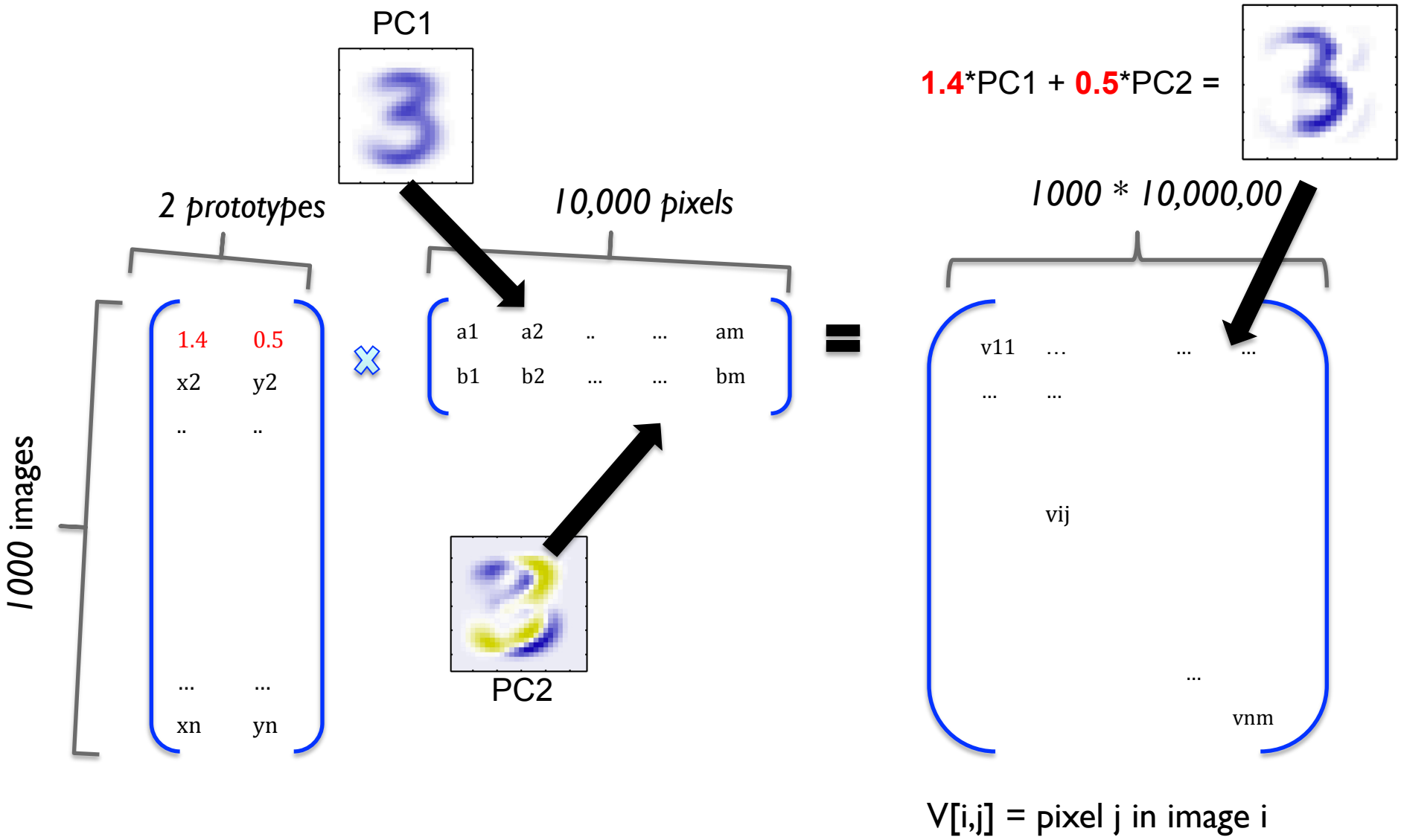
- Recap: PCA
- Directed models for text
  - Naïve Bayes
  - unsupervised Naïve Bayes (mixtures of multinomials)
  - probabilistic latent semantic indexing pLSI
    - and connection to PCA, LSI, MF
  - latent Dirichlet allocation (LDA)
    - inference with Gibbs sampling
- LDA-like models for graphs & etc

Quick Recap….

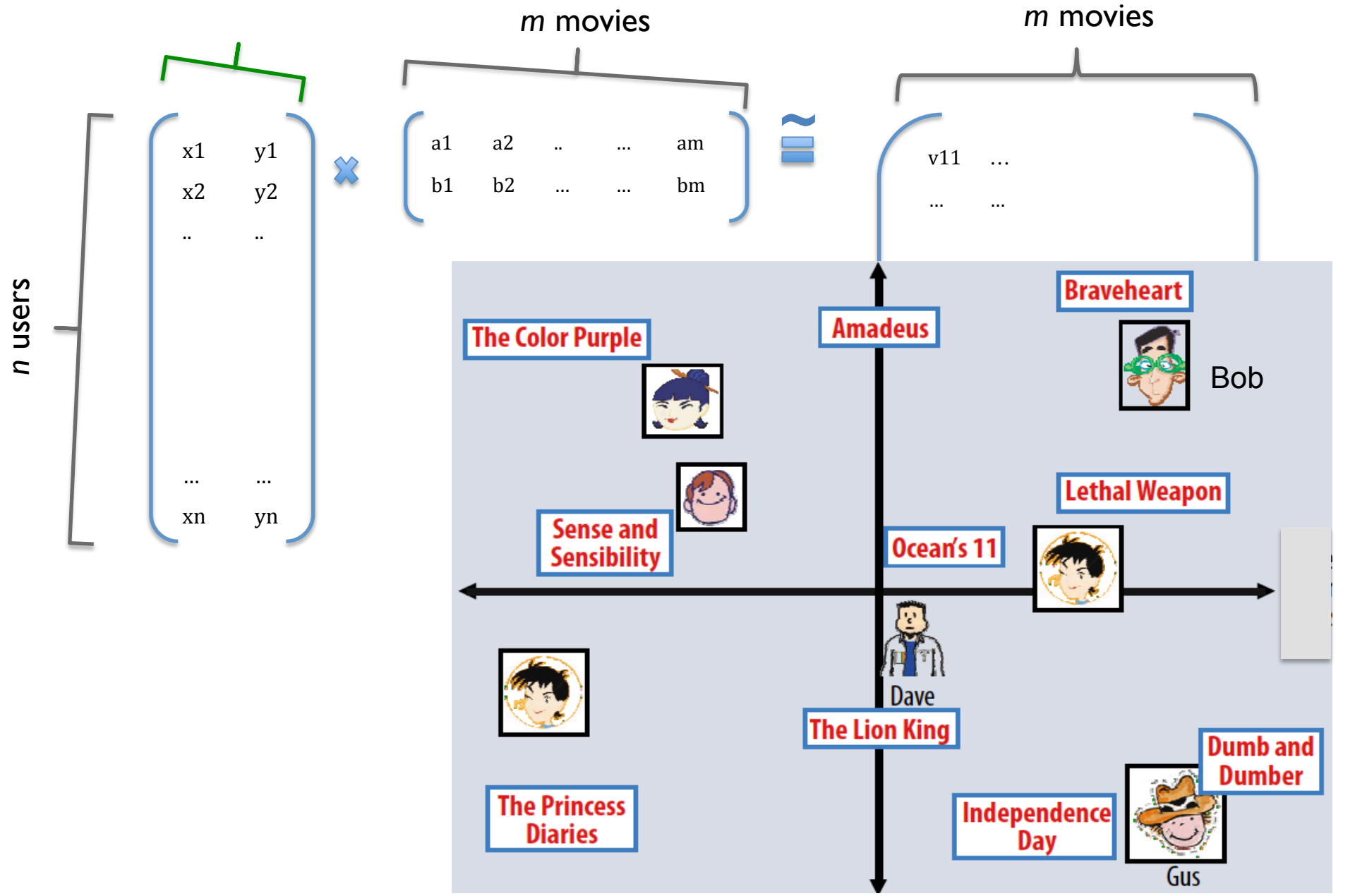
PCA as MF



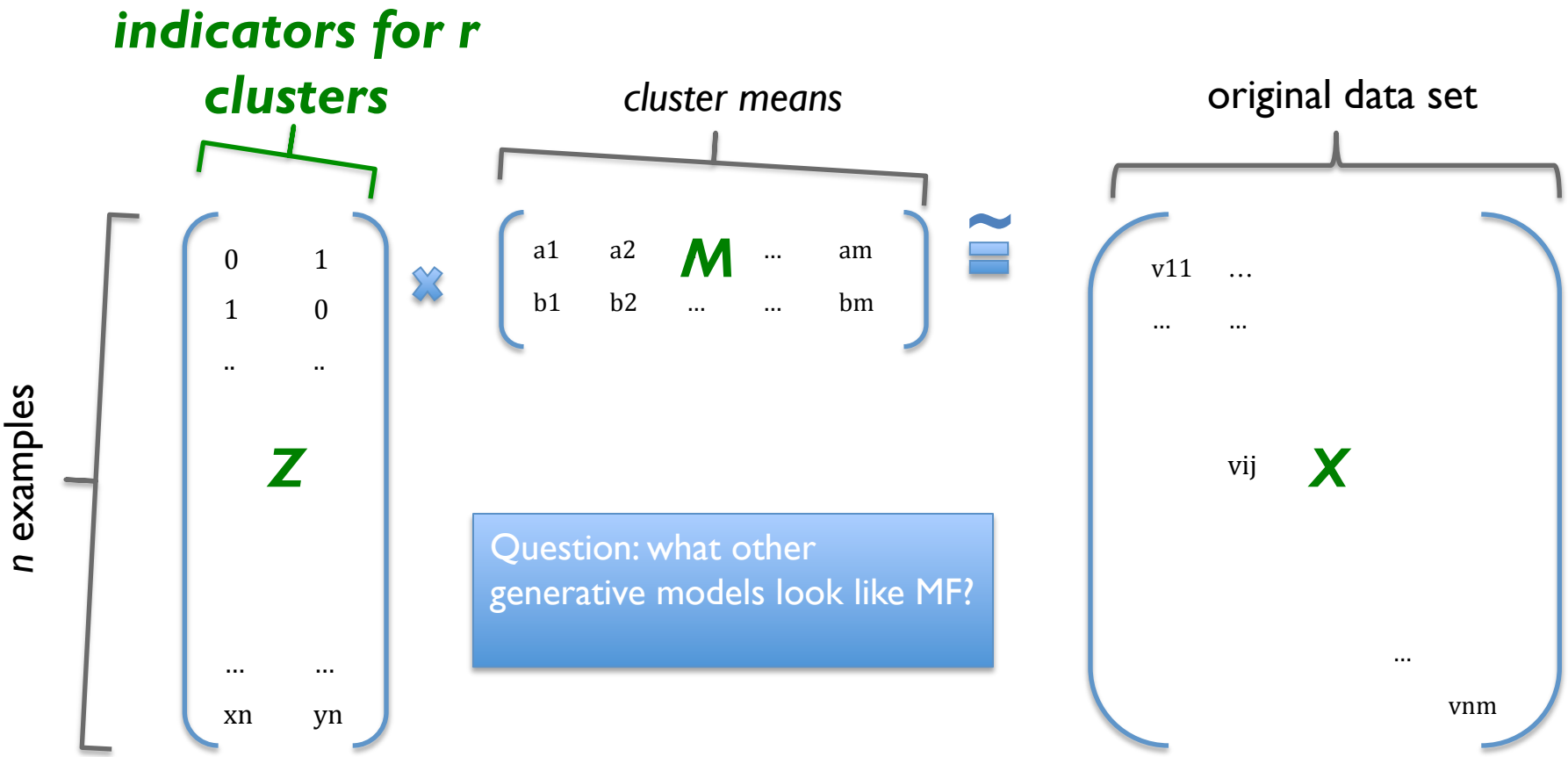
V[i,j] = pixel j in image i


PC1
1.4*PC1 + 0.5*PC2 =
2 prototypes
10,000 pixels
1000 * 10,000,00
1000 images
1.4 0.5
x2 y2
.. ..
... ...
xn yn
a1 a2 .. ... am
b1 b2 ... ... bm
PC2
v11 … ... ...
... ...
vij
...
vnm
V[i,j] = pixel j in image i

PCA for movie recommendation…



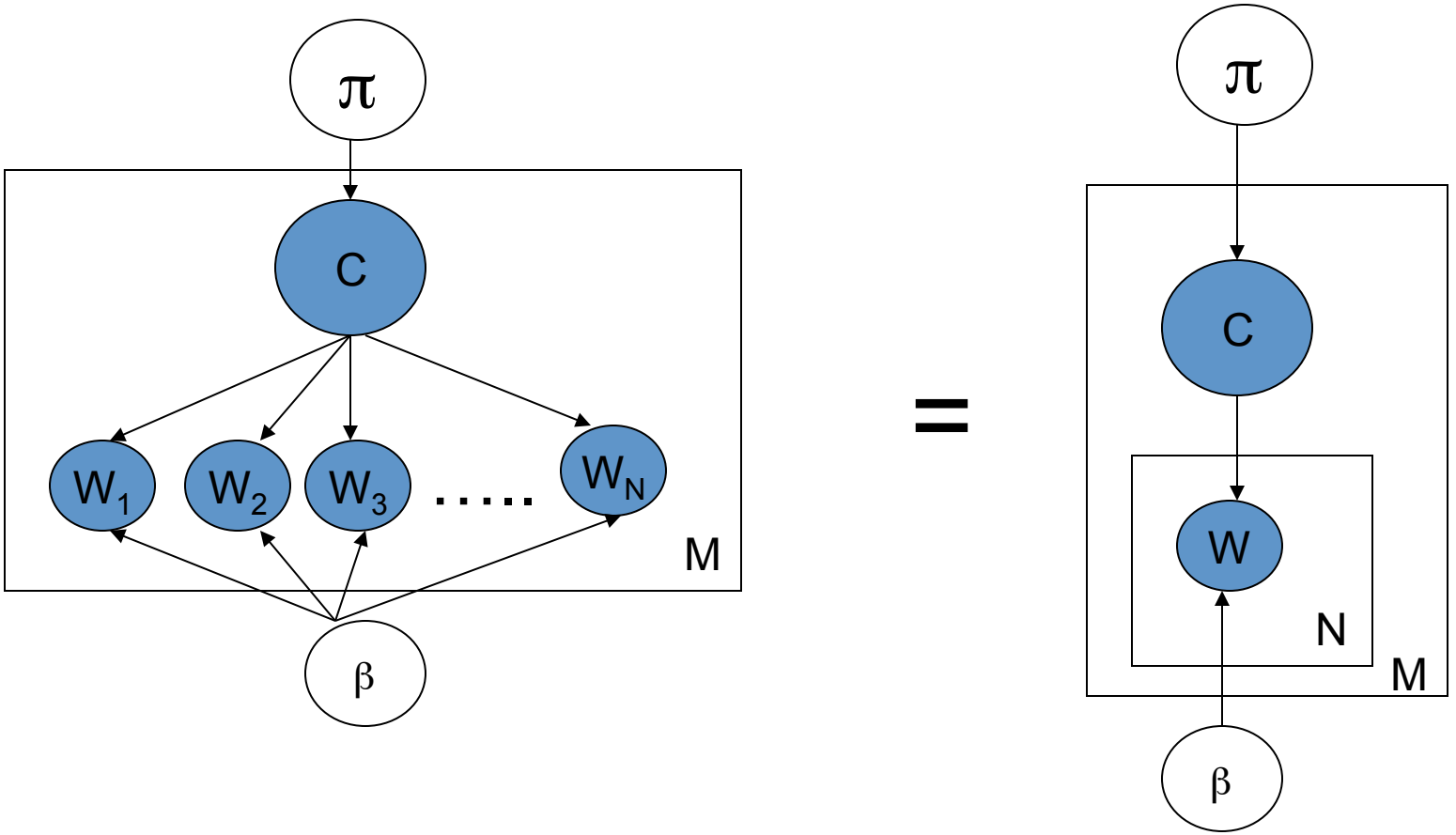
..... vs k-means


indicators for r clusters
cluster means
original data set
n examples
0 1
1 0
.. ..
Z
... ...
xn yn
a1 a2 M ... am
b1 b2 ... ... bm
v11 …
... ...
vij X
...
vnm
Question: what other generative models look like MF?

LDA AND OTHER DIRECTED MODELS FOR MODELING TEXT

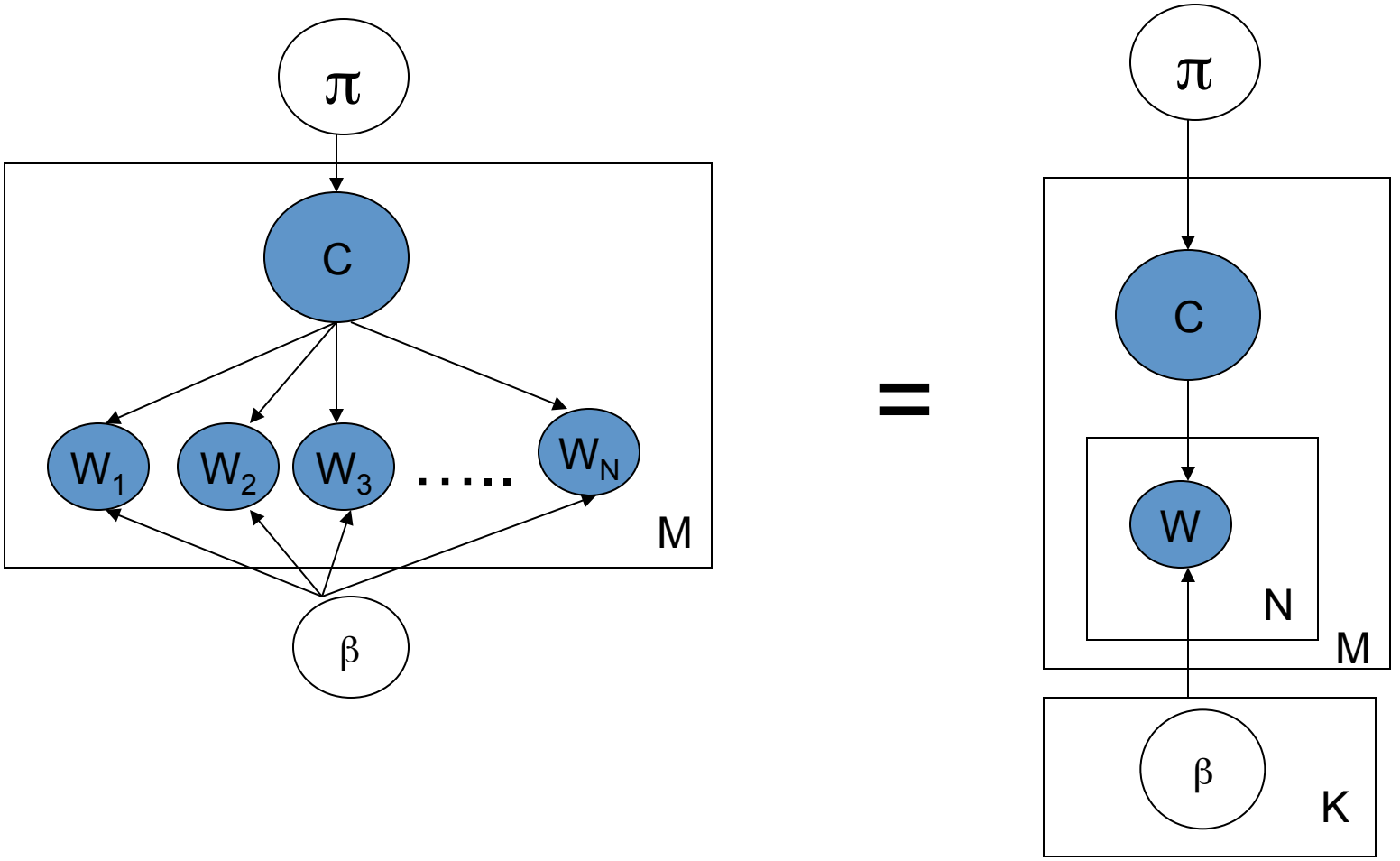
Supervised Multinomial Naïve Bayes

- Naïve Bayes Model: Compact notation



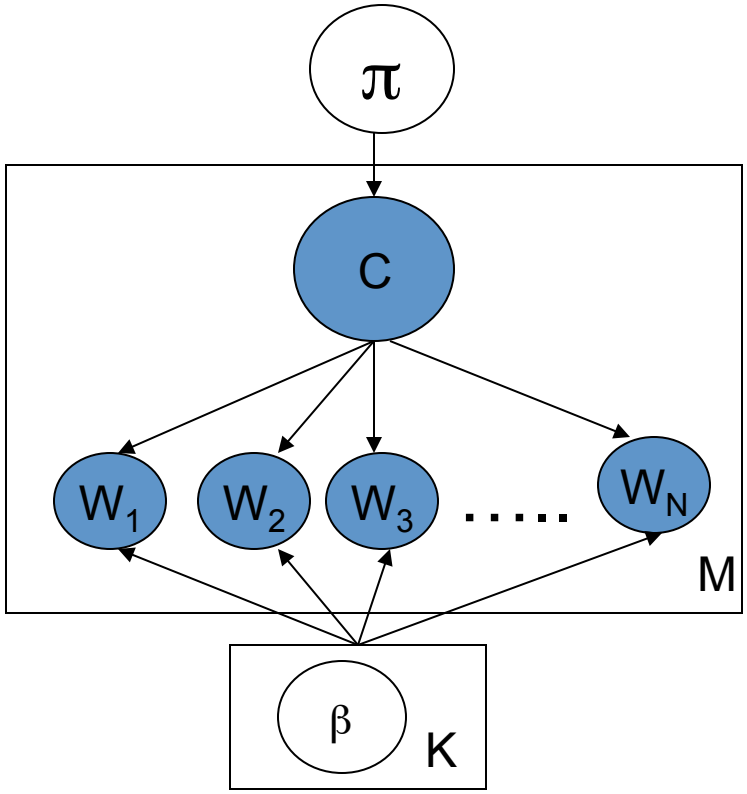
Supervised Multinomial Naïve Bayes

- Naïve Bayes Model: Compact representation



Supervised Naïve Bayes

- Multinomial Naïve Bayes

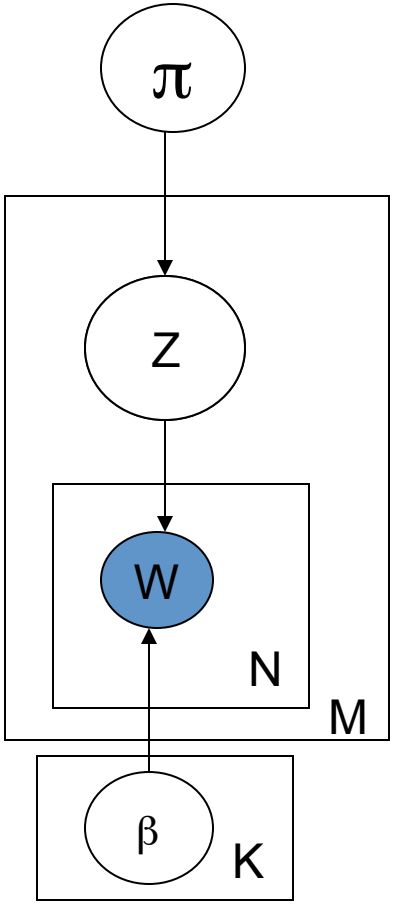


- For each class 1..K
  - Construct a multinomial β_i
- For each document d = 1,⋯, M
  - Generate C_d ~ Mult(. | π)
  - For each position n = 1,⋯, N_d
    - Generate w_n ~ Mult(.|β,C_d) …

  or if you prefer w_n ~ Pr(w|β_{Cd})

Unsupervised Naïve Bayes

- Mixture model: unsupervised naïve Bayes



• Joint probability of words and classes:

$$\prod_{d=1}^{M} P(w_1, \cdots, w_{N_d}, z_d | \beta, \pi) = \prod_{d=1}^{M} \left\{ \pi_{z_d} \prod_{n=1}^{N_d} \beta_{z_d, w_n} \right\}$$

$$\prod_{d=1}^{M} P(w_1, \cdots, w_{N_d} | \pi, \beta) = \prod_{d=1}^{N_d} \left\{ \sum_{k=1}^{K} \left(\pi_k \prod_{n=1}^{N_d} \beta_{k, w_n} \right) \right\}$$

Solved using EM

Unsupervised Naïve Bayes

$$\log\left(\sum_{k=1}^{K}\left[\pi_k\prod_{n=1}^{N_d}\beta_{k,w_n}\right]\right) \geq \sum_{k=1}^{K}\left\{\gamma_k\log\left(\pi_k\prod_{n=1}^{N_d}\beta_{k,w_n}\right)\right\}+H(\gamma)$$

- Mixture model: EM solution

E-step:

$$\gamma_{dk}^{(t+1)} = \frac{\pi_k^{(t)}\prod_{n=1}^{N_d}\beta_{k,w_n}^{(t)}}{\sum_k \pi_k^{(t)}\prod_{n=1}^{N_d}\beta_{k,w_n}^{(t)}}$$

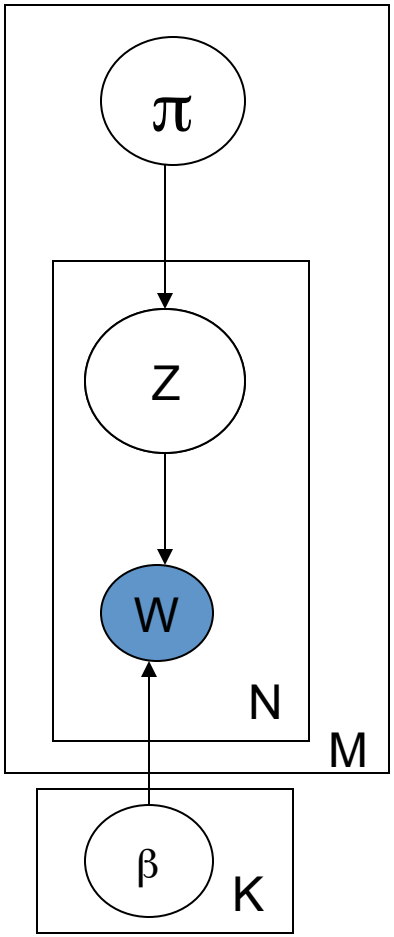
M-step:

$$\begin{cases} \pi_k^{(t+1)} = \dfrac{\sum_{d=1}^{M}\gamma_{dk}^{(t)}}{M} \\ \beta_{k,w}^{(t+1)} = \dfrac{\sum_{d=1}^{M}\gamma_{dk}^{(t)}n(d,w)}{\sum_{d=1}^{M}\gamma_{nk}^{(t)}\sum_w n(d,w)} \end{cases}$$

Key capability: estimate distribution of **latent variables** given **observed variables**

Beyond Naïve Bayes - Probabilistic Latent Semantic Indexing (PLSI)

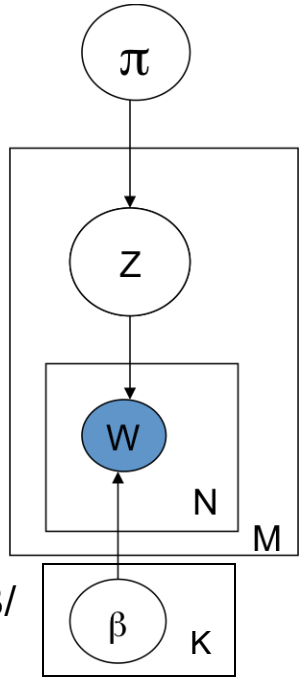
- Every document is a mixture of topics



- For i=1…K:
  - Let β_i be a multinomial over words
- For each document *d*:
  - Let π_d be a distribution over {1,..,K}
  - For each word position in *d:*
    - Pick a topic *z* from π_d
    - Pick a word *w* from β_i

Latent variable for *each word* not for *each document*

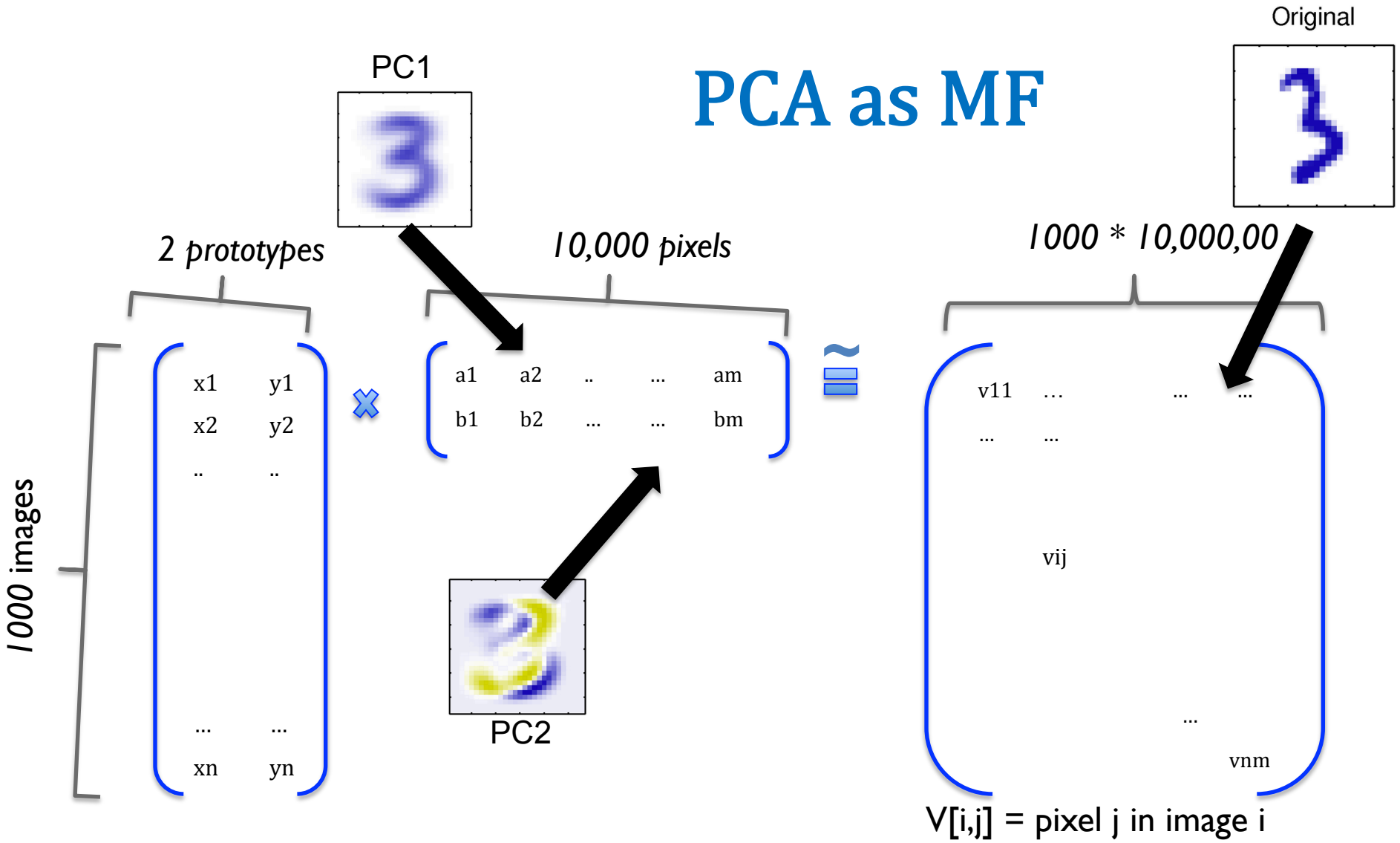
Can still learn with EM



compare to unsupervised NB/ mixture of multinomials

LATENT SEMANTIC INDEXING
VS
SINGULAR VALUE DECOMPOSITION
VS
PRINCIPLE COMPONENTS ANALYSIS

PCA as MF

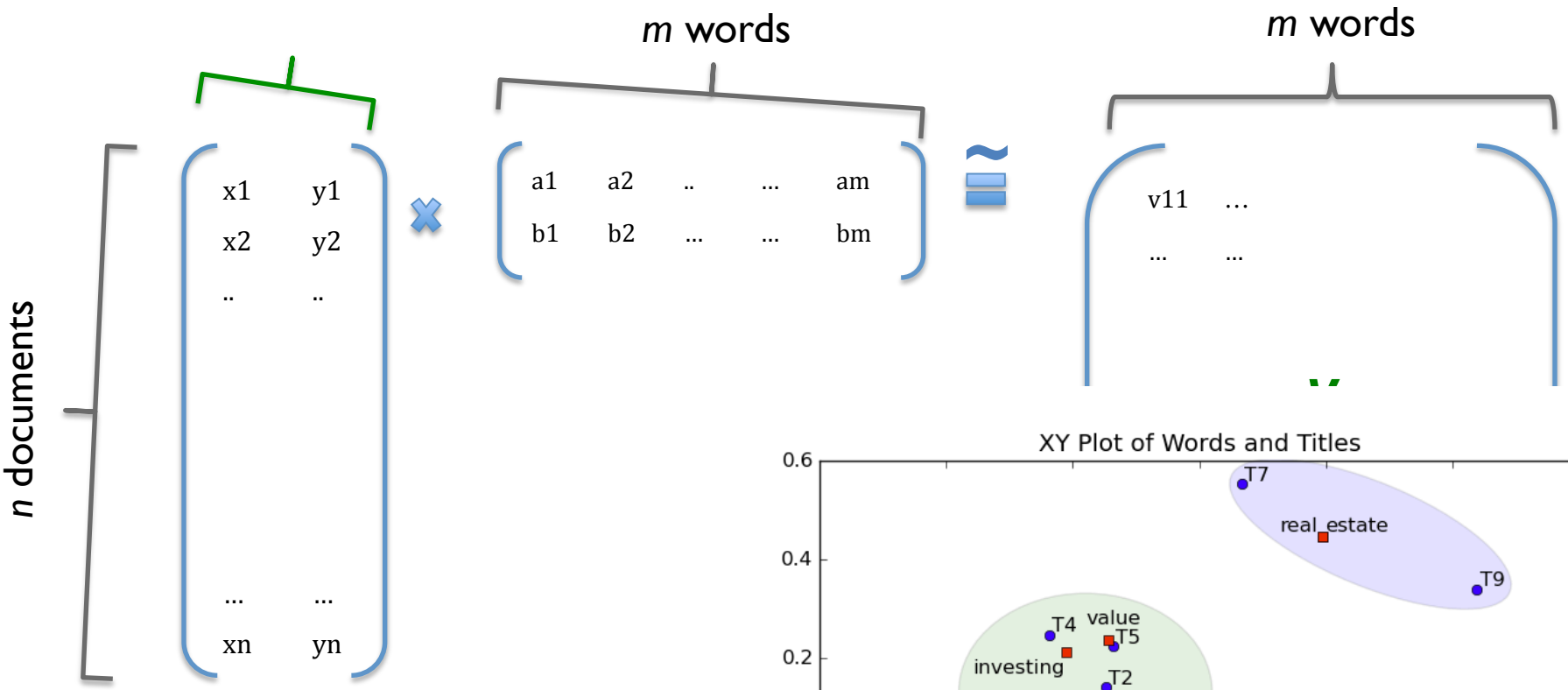


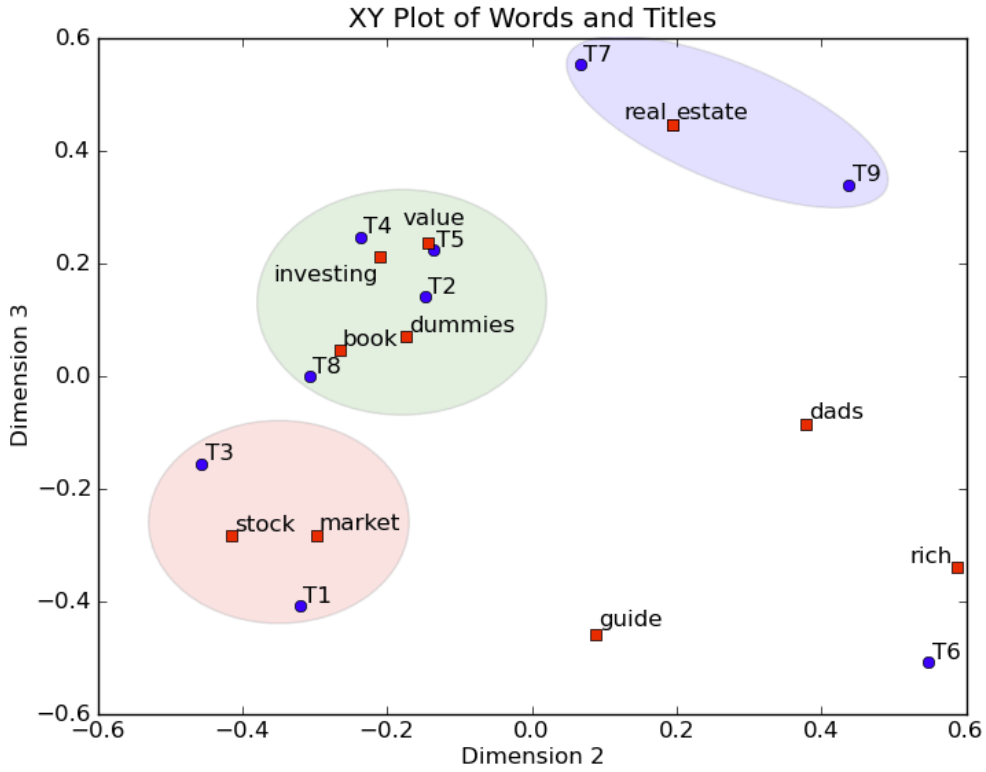
Remember C_X – covariance of features of the original data matrix?
We can look also at C_Z and see what the covariance is…
Turns out $C_Z[i.j]=0$ for i!=j and $C_Z[i,j]= \lambda_i$

Background: PCA vs LSI

- Some connections
  - PCA is closely related to *singular value decomposition* (SVD): see handout
    - we factored data matrix X into Z*V, V is eigenvectors of C_X
    - *one more step*: factor Z into U*C where Σ is diagonal and $\Sigma(i,i)=\text{sqrt}(\lambda_i)$ and variables in U have unit variance
    - then $X = U*\Sigma*V$ and this is called SVD
- When X is a term-document matrix this is called *latent semantic indexing* (LSI)

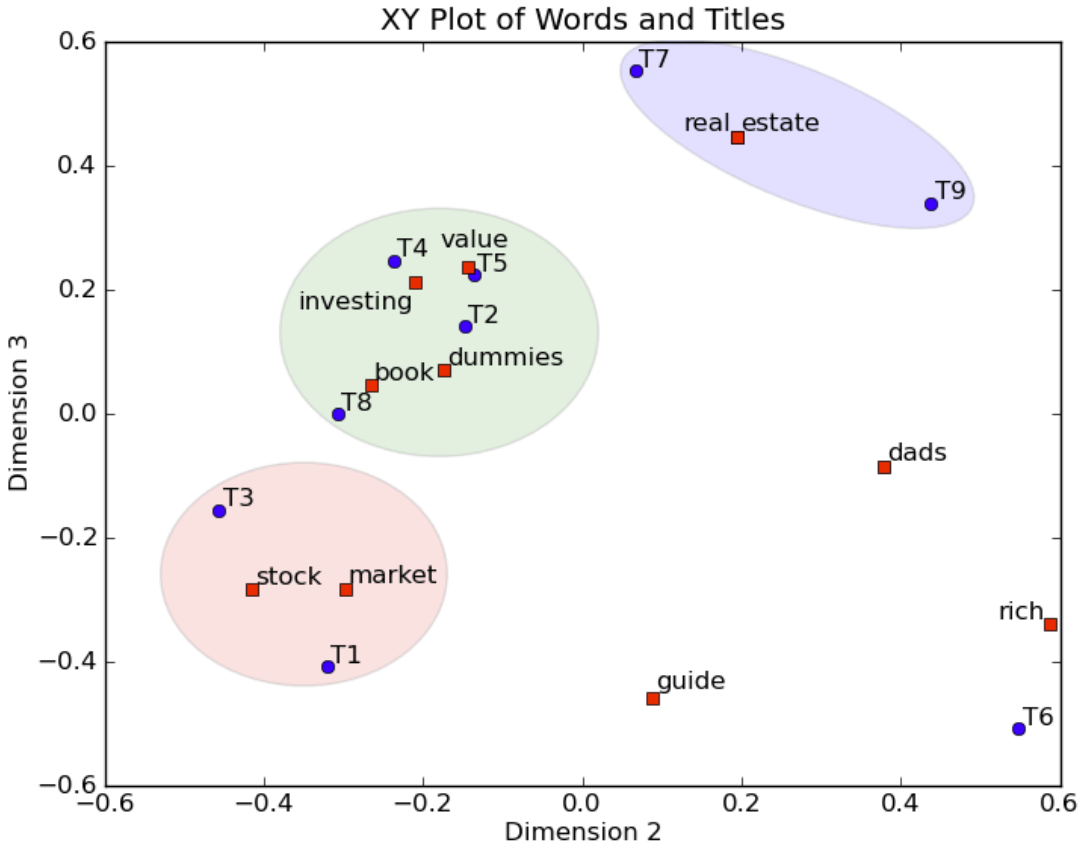
LSI


m words
m words
n documents
x1 y1
x2 y2
.. ..
... ...
xn yn
a1 a2 .. ... am
b1 b2 ... ... bm
v11 …
... ...


XY Plot of Words and Titles
Dimension 2
Dimension 3
T7
real_estate
T9
T4
value
T5
investing
T2
dummies
book
T8
dads
T3
stock
market
rich
T1
guide
T6
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

LSI

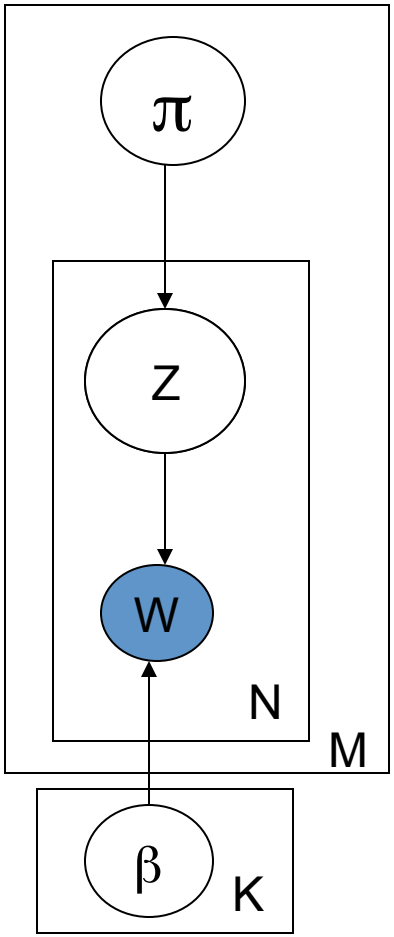
i. The Neatest Little Guide to Stock Market Investing
ii. Investing For Dummies, 4th Edition
iii. The Little Book of Common Sense Investing: The Only Way to Guarantee Your Fair Share of Stock M
iv. The Little Book of Value Investing
v. Value Investing: From Graham to Buf
vi. Rich Dad's Guide to Investing: What t and the Middle Class Do Not!
vii. Investing in Real Estate, 5th Edition
viii. Stock Investing For Dummies
ix. Rich Dad's Advisors: The ABC's of Re Finding Hidden Profits Most Investor



http://technowiki.wordpress.com/2011/08/27/latent-semantic-analysis-lsa-tutorial/

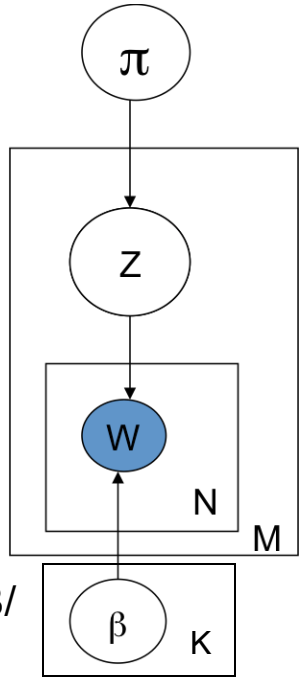
Beyond Naïve Bayes - Probabilistic Latent Semantic Indexing (PLSI)

- Every document is a mixture of topics



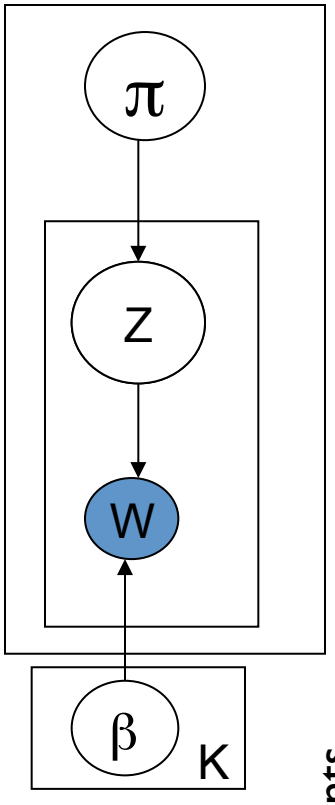
- For i=1…K:
  - Let β_i be a multinomial over words
- For each document *d*:
  - Let π_d be a distribution over {1,..,K}
  - For each word position in *d:*
    - Pick a topic *z* from π_d
    - Pick a word *w* from β_i

Latent variable for *each word* not for *each document*



compare to unsupervised NB/ mixture of multinomials

Beyond Naïve Bayes - Probabilistic Latent Semantic Indexing (PLSI)



- For i=1…K:
  - Let β_i be a multinomial over words
- For each document *d*:
  - Let π_d be a distribution over {1,..,K}
  - For each word position in *d:*
    - Pick a topic *z* from π_d
    - Pick a word *w* from β_i

m words

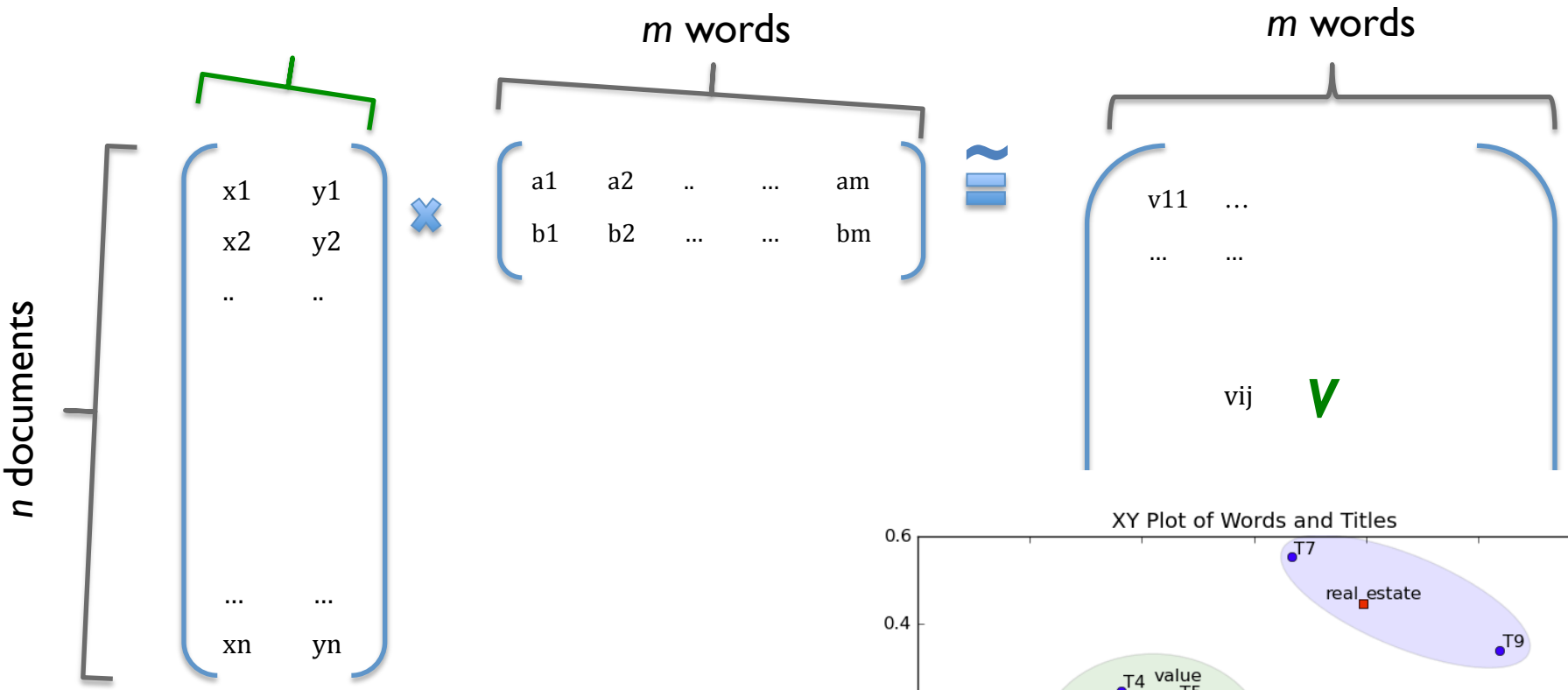
β_1	a1	a2	..	...	am
β_2	b1	b2	...	...	bm

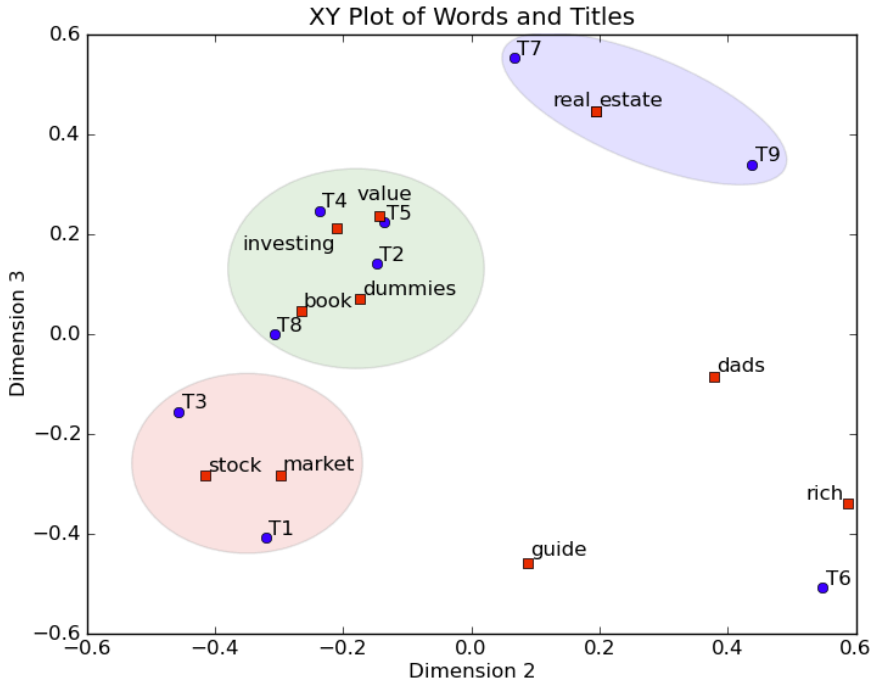
n documents

π_1	x1	y1
π_2	x2	y2
..	..	..
...	...	...
π_n	xn	yn

$$\Pr[X(i,j)] = \sum_k \Pr(Z = k \mid \pi[i,k]) * \Pr(W = j \mid \beta[k,j])$$

LSI or pLSI


m words
m words
n documents
x1 y1
x2 y2
.. ..
... ...
xn yn
a1 a2 .. ... am
b1 b2 ... ... bm
v11 …
... ...
vij
V


XY Plot of Words and Titles
Dimension 2
Dimension 3
T7
real_estate
T9
T4
value
T5
investing
T2
book
dummies
T8
dads
T3
stock
market
T1
rich
guide
T6

Background: LSI vs NMF

- http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf

PLSI and MF and Nonnegative MF

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Computational Statistics and Data Analysis 52 (2008) 3913–3927

COMPUTATIONAL STATISTICS & DATA ANALYSIS

www.elsevier.com/locate/csda

On the equivalence between Non-negative Matrix Factorization and Probabilistic Latent Semantic Indexing

Chris Ding[a], Tao Li[b,*], Wei Peng[b]

PLSI and MF and Nonnegative MF

The general form of NMF is

$$F \cong CH^{\mathrm{T}},$$

We considered minimized **L2 reconstruction error.**

Another choice: constrain C, H to be non-negative and minimize J_{NMF}:

$$J_{\mathrm{NMF}} = \sum_{i=1}^{m}\sum_{j=1}^{n} F_{ij} \log \frac{F_{ij}}{(CH^{\mathrm{T}})_{ij}} - F_{ij} + (CH^{\mathrm{T}})_{ij}.$$

PLSI maximizes the likelihood

$$\max J_{\mathrm{PLSI}}, \qquad J_{\mathrm{PLSI}} = \sum_{i=1}^{m}\sum_{j=1}^{n} F_{ij} \log P(w_i, d_j)$$

where

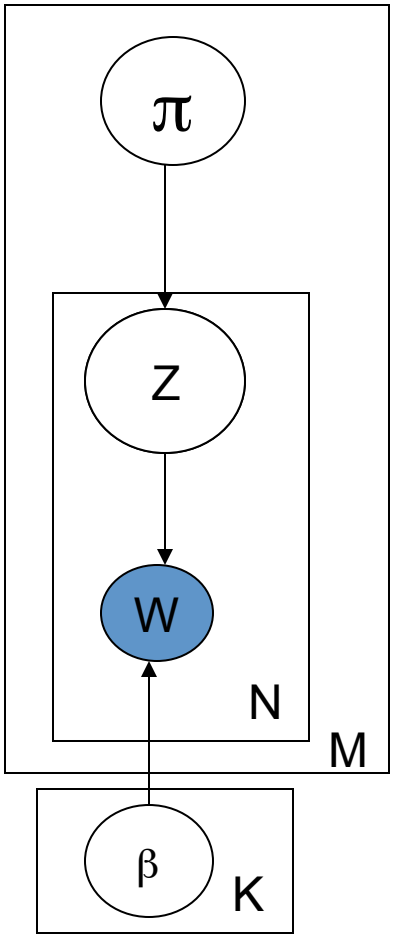
$$P(w_i, d_j) = \sum_k P(w_i, d_j|z_k)P(z_k)$$

$$= \sum_k P(w_i|z_k)P(d_j|z_k)P(z_k),$$

Ding et al: the objective functions are for NMF and PLSI are equivalent

Beyond Naïve Bayes - Probabilistic Latent Semantic Indexing (PLSI)

- Every document is a mixture of topics



- For i=1…K:
  - Let β_i be a multinomial over words
- For each document *d*:
  - Let π_d be a distribution over {1,..,K}
  - For each word position in *d:*
    - Pick a topic *z* from π_d
    - Pick a word *w* from β_i

- **Turns out to be hard to fit:**
  - **Lots of parameters!**
  - **Also: only applies to the training data**

LATENT DIRICHLET ANALYSIS (LDA)

The LDA Topic Model

Latent Dirichlet Allocation

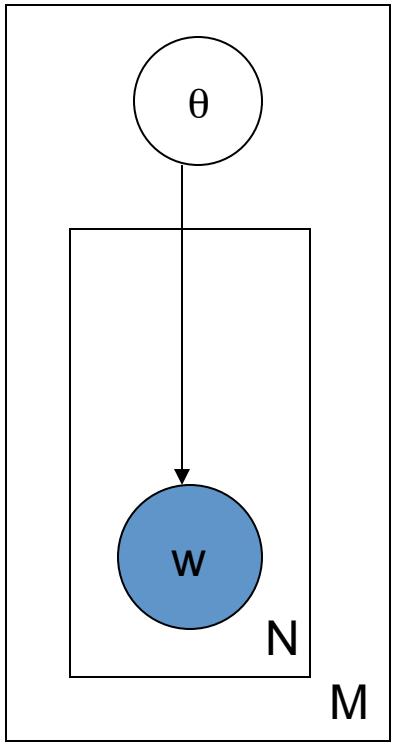
David M. Blei
Computer Science Division
University of California
Berkeley, CA 94720, USA

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

LDA

- Motivation



Assumptions: 1) documents are i.i.d 2) *within* a document, words are i.i.d. (bag of words)

•For each document d = 1,⋯,M

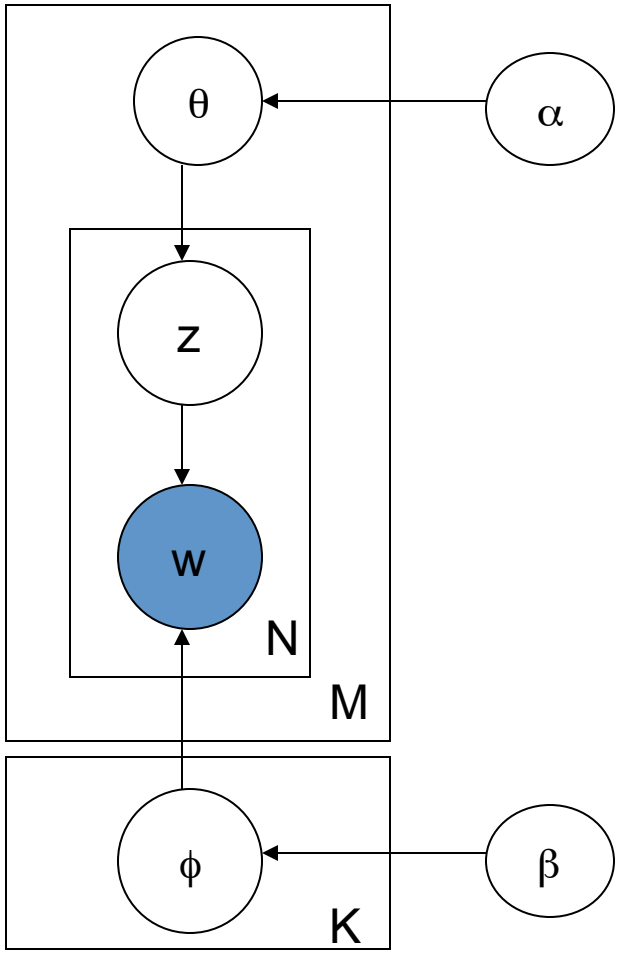
- Generate $\theta_d \sim D_1(\ldots)$
- For each word n = 1,⋯, N_d
  - •generate $w_n \sim D_2(\ . \mid \vartheta_{d_n})$

Now pick your favorite distributions for D_1, D_2

$$\Pr(z = j \mid n_1, n_{2,} \ldots, n_k, \alpha) = \frac{n_j + \alpha_j}{n_1 + \alpha_1 + \ldots n_k + \alpha_k}$$

- Latent Dirichlet Allocation

"Mixed membership"



- For each document d = 1,⋯,M
  - Generate θ_d ~ Dir(. | α)
  - For each position n = 1,⋯, N_d
    - generate z_n ~ Mult(. | θ_d)
    - generate w_n ~ Mult(. | β_{z_n})

$$\prod_{d=1}^{N_d} P(w_1, \cdots, w_{N_d} | \beta, \alpha)$$

$$= \prod_{d=1}^{N_d} \int_{\theta_d} P(\theta_d | \alpha) \left\{ \prod_{n=1}^{N_d} \left(\sum_k \theta_{dk} \beta_{k w_n} \right) \right\} d\theta_d$$

- LDA's view of a document

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.

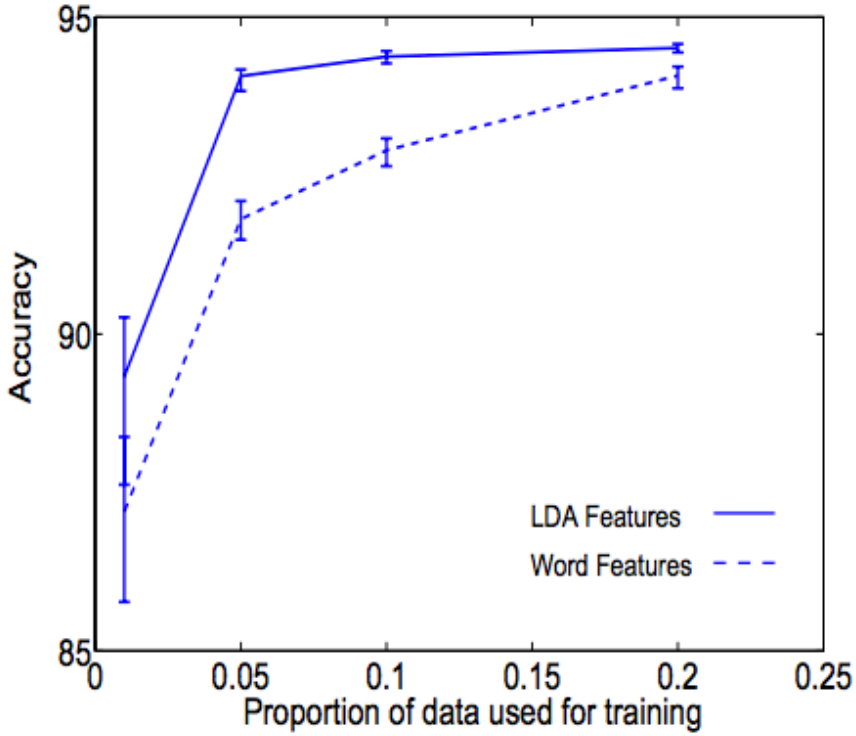
"Arts" "Budgets" "Children" "Education"

- LDA topics

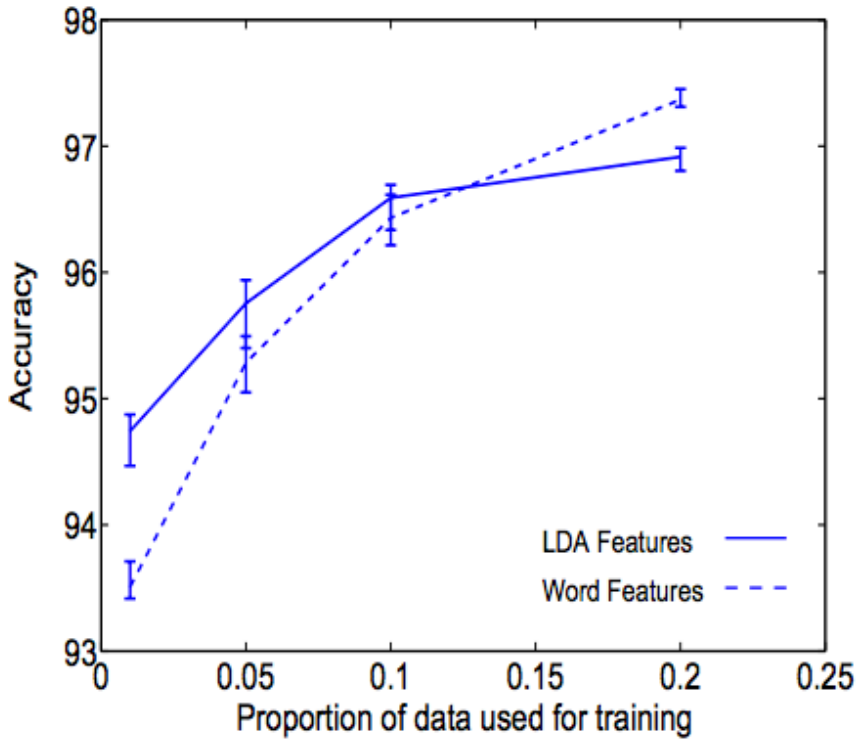
"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY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

LDA used as dimension reduction for classification

50 topics vs all words, SVM



(a)



(b)

Figure 10: Classification results on two binary classification problems from the Reuters-21578 dataset for different proportions of training data. Graph (a) is EARN vs. NOT EARN. Graph (b) is GRAIN vs. NOT GRAIN.

LDA used for CF

Users rating 100+ movies only

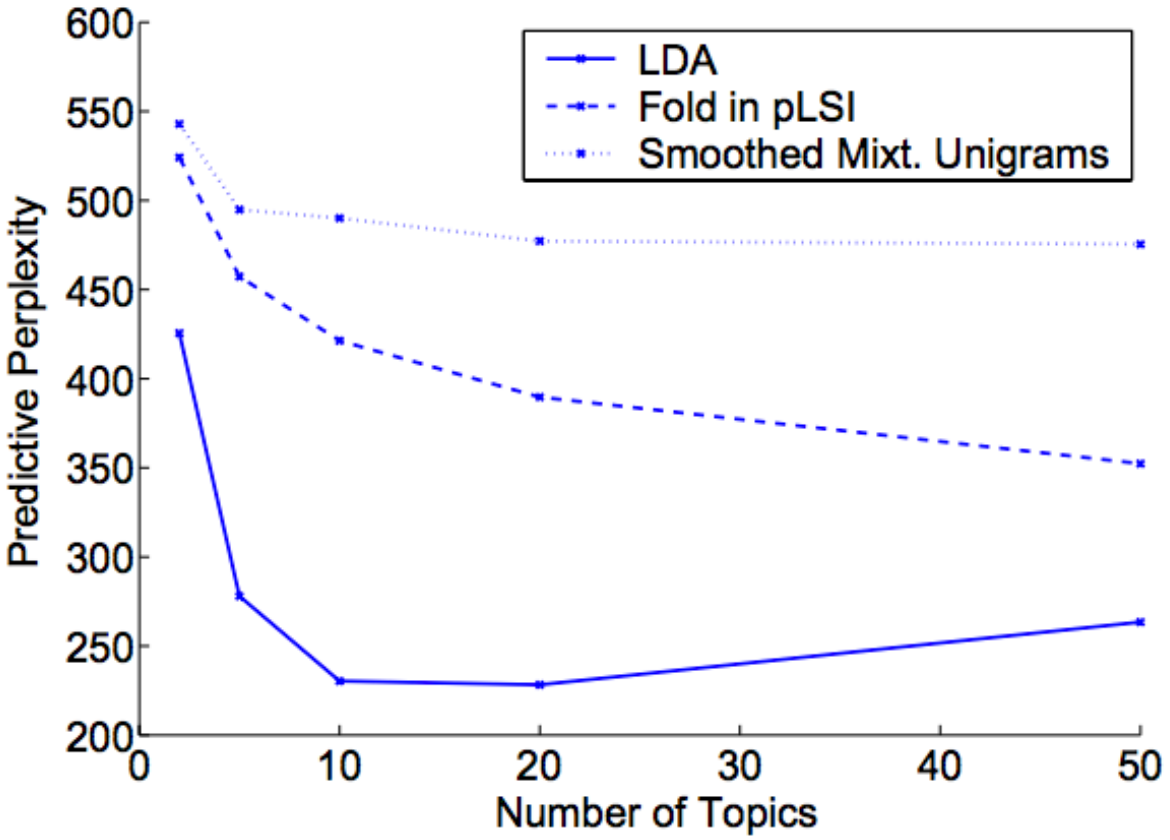


Figure 11: Results for collaborative filtering on the EachMovie data.

LDA

- Latent Dirichlet Allocation
  - Parameter learning:
    - Variational EM
      - Numerical approximation using lower-bounds
      - Results in biased solutions
      - Convergence has numerical guarantees
    - Gibbs Sampling
      - Stochastic simulation
      - unbiased solutions
      - Stochastic convergence

LDA

- 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.

Why does Gibbs sampling work?

- What's the fixed point?
  - Stationary distribution of the chain is the joint distribution
- When will it converge (in the limit)?
  - If graph defined by the chain is connected
- How long will it take to converge?
  - Depends on second eigenvector of that graph

⊡ initialisation
zero all count variables, $n_m^{(k)}, n_m, n_k^{(t)}, n_k$
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: $n_k^{(t)} + 1$
    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: $n_m^{(k)} - 1; n_m - 1; n_k^{(t)} - 1; n_k - 1$
      sample topic index $\tilde{k} \sim p(z_i|\vec{z}_{\neg i}, \vec{w})$
      ⊡ use the new assignment of $z_{m,n}$ to the term t for word $w_{m,n}$ to:
      increment counts and sums: $n_m^{(\tilde{k})} + 1; n_m + 1; n_{\tilde{k}}^{(t)} + 1; n_{\tilde{k}} + 1$
    end for
  end for
  ⊡ check convergence and read out parameters

$$p(z_i{=}k|\vec{z}_{\neg i}, \vec{w}) = \frac{p(\vec{w}, \vec{z})}{p(\vec{w}, \vec{z}_{\neg i})} = \frac{p(\vec{w}|\vec{z})}{p(\vec{w}_{\neg i}|\vec{z}_{\neg i})p(w_i)} \cdot \frac{p(\vec{z})}{p(\vec{z}_{\neg i})}$$

$$\propto \frac{\Delta(\vec{n}_z + \vec{\beta})}{\Delta(\vec{n}_{z,\neg i} + \vec{\beta})} \cdot \frac{\Delta(\vec{n}_m + \vec{\alpha})}{\Delta(\vec{n}_{m,\neg i} + \vec{\alpha})}$$

$$\propto \frac{\Gamma(n_k^{(t)} + \beta_t)\,\Gamma(\sum_{t=1}^{V} n_{k,\neg i}^{(t)} + \beta_t)}{\Gamma(n_{k,\neg i}^{(t)} + \beta_t)\,\Gamma(\sum_{t=1}^{V} n_k^{(t)} + \beta_t)} \cdot \frac{\Gamma(n_m^{(k)} + \alpha_k)\,\Gamma(\sum_{k=1}^{K} n_{m,\neg i}^{(k)} + \alpha_k)}{\Gamma(n_{m,\neg i}^{(k)} + \alpha_k)\,\Gamma(\sum_{k=1}^{K} n_m^{(k)} + \alpha_k)}$$

$$\propto \frac{n_{k,\neg i}^{(t)} + \beta_t}{\sum_{t=1}^{V} n_{k,\neg i}^{(t)} + \beta_t} \cdot \frac{n_{m,\neg i}^{(t)} + \alpha_k}{[\sum_{k=1}^{K} n_m^{(k)} + \alpha_k] - 1}$$

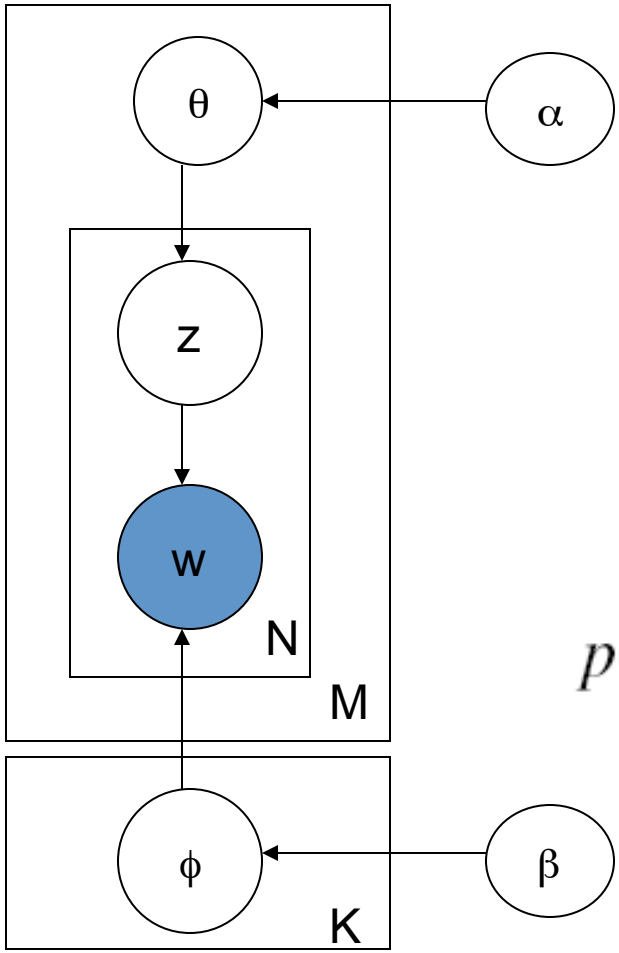
Called “collapsed Gibbs sampling” since you’ve marginalized away some variables

Fr*: Parameter estimation for text analysis* - Gregor Heinrich

LDA

- Latent Dirichlet Allocation

"Mixed membership"



- Randomly initialize each $z_{m,n}$
- Repeat for t=1,….
  - For each doc *m,* word *n*
    - Find Pr(z_{mn}=*k*|other z's)
    - Sample z_{mn} according to that distr.

$$p(z_i=k|\vec{z}_{\neg i},\vec{w}) =$$

$$\propto \frac{n^{(t)}_{k,\neg i}+\beta_t}{\sum_{t=1}^{V} n^{(t)}_{k,\neg i}+\beta_t} \cdot \frac{n^{(t)}_{m,\neg i}+\alpha_k}{[\sum_{k=1}^{K} n^{(k)}_{m}+\alpha_k]-1}$$

EVEN MORE DETAIL ON LDA…

Way way more detail

```
# 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
```

```
    # 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


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
```

```
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)

def flip(self, d, j, k_old, k_new):
    """update counts to reflect a changed value of z[d][j]"""
    if k_old != k_new:
        w = self.x[d][j]
        self.docTopicCount[d].add(k_old, -1)
        self.wordTopicCount[w].add(k_old, -1)
        self.wordTopicCount[w].add(k_new, +1)
        self.totalTopicCount.add(k_old, -1)
        self.totalTopicCount.add(k_new, +1)
        self.z[d][j] = k_new
```

```
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)


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
```

What gets learned.....

```
def phi(self,w,k):
    """weight of word w under topic k"""
    num = (self.wordTopicCount[w][k] + self.beta)
    denom = (self.totalTopicCount[k] + self.totalWords * self.beta)
    return num/denom

def theta(self,d,k):
    """weight of doc unde
    num = (self.docTopicC
    denom = (sum(self.doc'
    return num/denom
```

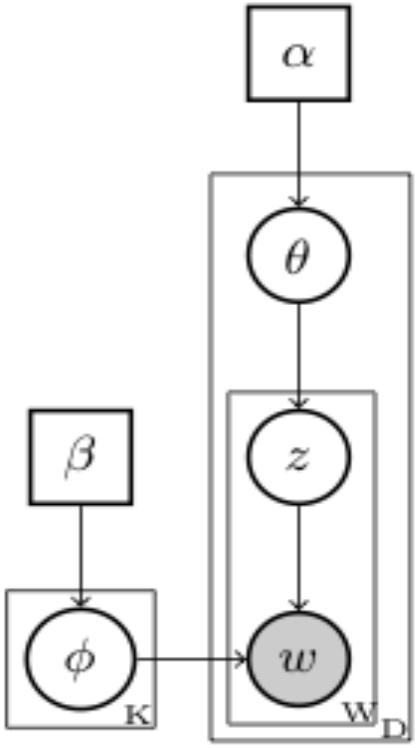


Figure 1: Graphical model for LDA.

In A Math-ier Notation

```
# 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
```

N[,k]*

N[d,k]

N[,*]=V*

M[w,k]

for each document *d* and word position *j* in *d*

- *z[d,j]* = *k*, a random topic
- *N[d,k]*++
- *W[w,k]*++ where *w* = id of *j*-th word in *d*

```
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
```

```
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)
```

for each pass t=1,2,….

for each document *d* and word position *j* in *d*

- *z[d,j] = k,* a new random topic
- update *N, W* to reflect the new assignment of z:
  - *N[d,k]++; N[d,k'] - -* where *k'* is old *z[d,j]*
  - *W[w,k]++; W[w,k'] - -* where *w* is *w[d,j]*

```
def flip(self, d, j, k_old, k_new):
    """update counts to reflect a changed value of z[d][j]"""
    if k_old != k_new:
        w = self.x[d][j]
        self.docTopicCount[d].add(k_old, -1)
        self.wordTopicCount[w].add(k_old, -1)
        self.wordTopicCount[w].add(k_new, +1)
        self.totalTopicCount.add(k_old, -1)
        self.totalTopicCount.add(k_new, +1)
        self.z[d][j] = k_new
```

$$p(Z_{d,j}=k\,|\,..) \propto \Pr(Z_{d,j}=k\,|\,"d") * \Pr(W_{d,k}=w\,|\,Z_{d,j}=k,...)$$

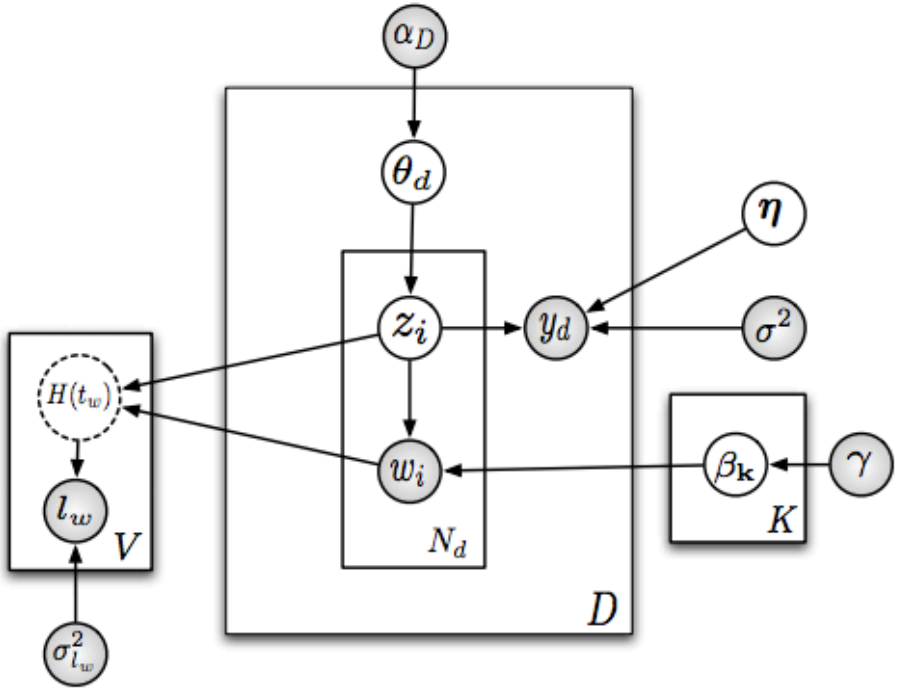
$$= \frac{N[k,d]-C_{d,j,k}+\alpha}{Z} \cdot \frac{W[w,k]-C_{d,j,k}+\beta}{(W[*,k]-C_{d,j,k})+\beta N[*,*]}$$

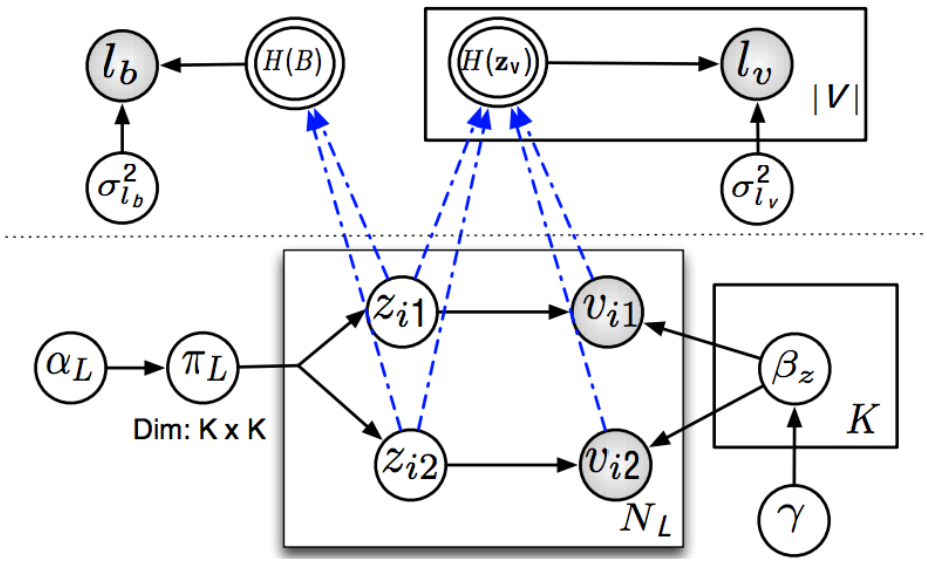
$$C_{d,j,k} = \begin{cases} 1 & Z_{d,j}=k \\ 0 & \text{else} \end{cases}$$

```
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
```

Some comments on LDA

- Very widely used model
- Also a component of many other models



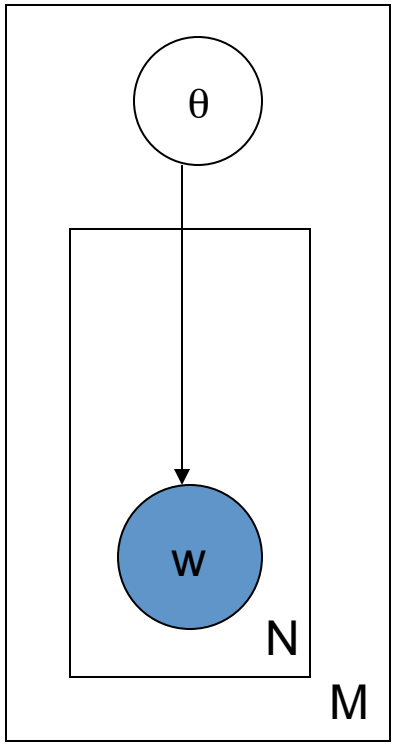


Techniques for Modeling Graphs

Latent Dirichlet Allocation++

Review - LDA

- Motivation



Assumptions: 1) documents are i.i.d 2) *within* a document, words are i.i.d. (bag of words)

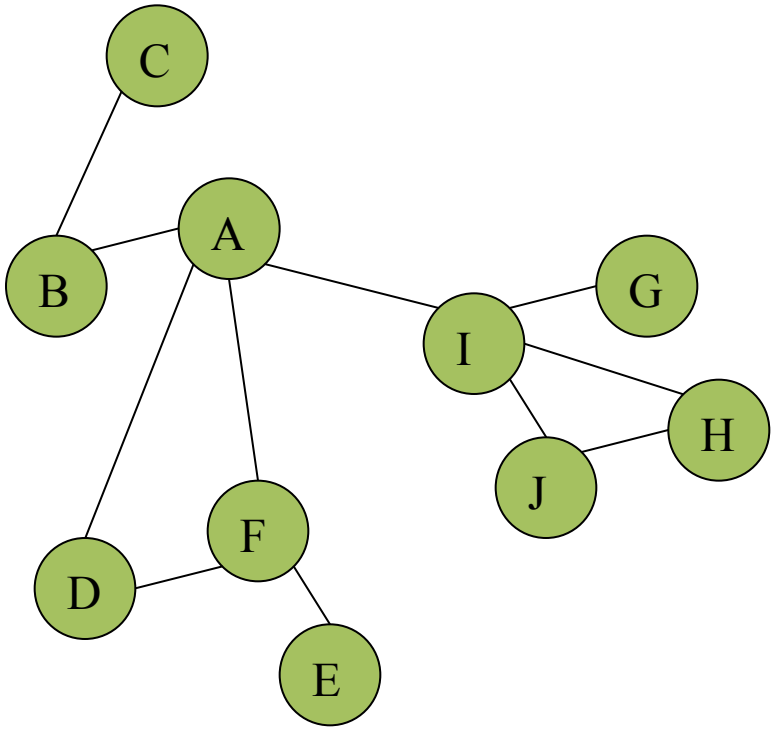
•For each document d = 1,⋯,M

- Generate $\theta_d \sim D_1(\ldots)$
- For each word n = 1,⋯, N_d
  - •generate $w_n \sim D_2(\phi \mid \vartheta_{d_n})$

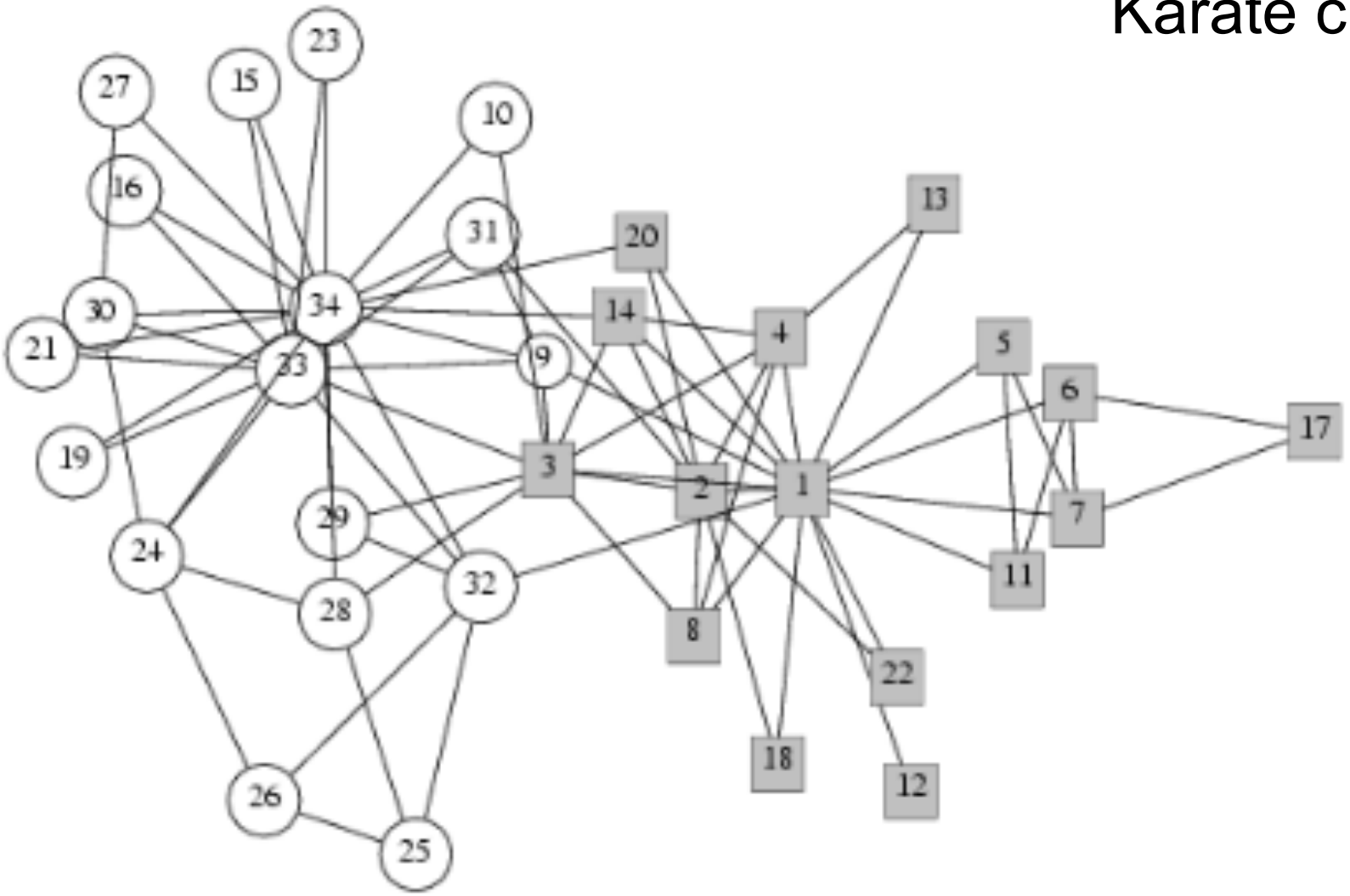
Docs and words are *exchangeable*.

A Matrix Can Also Represent a Graph

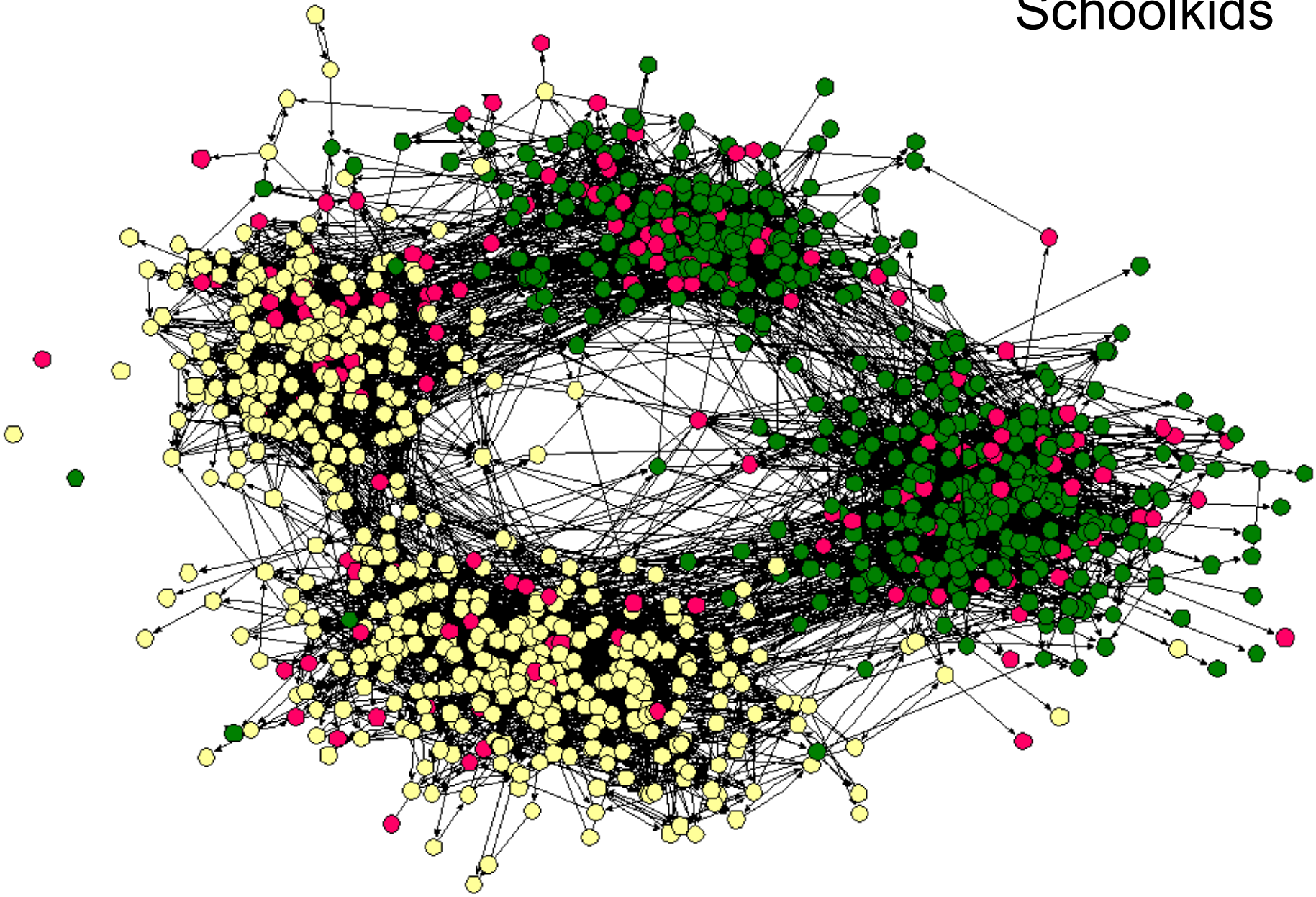
	A	B	C	D	E	F	G	H	I	J
A		1		1		1				
B	1		1							
C		1								
D	1					1				
E						1				
F	1			1	1					
G									1	
H							1		1	1
I							1	1		1
J								1	1	

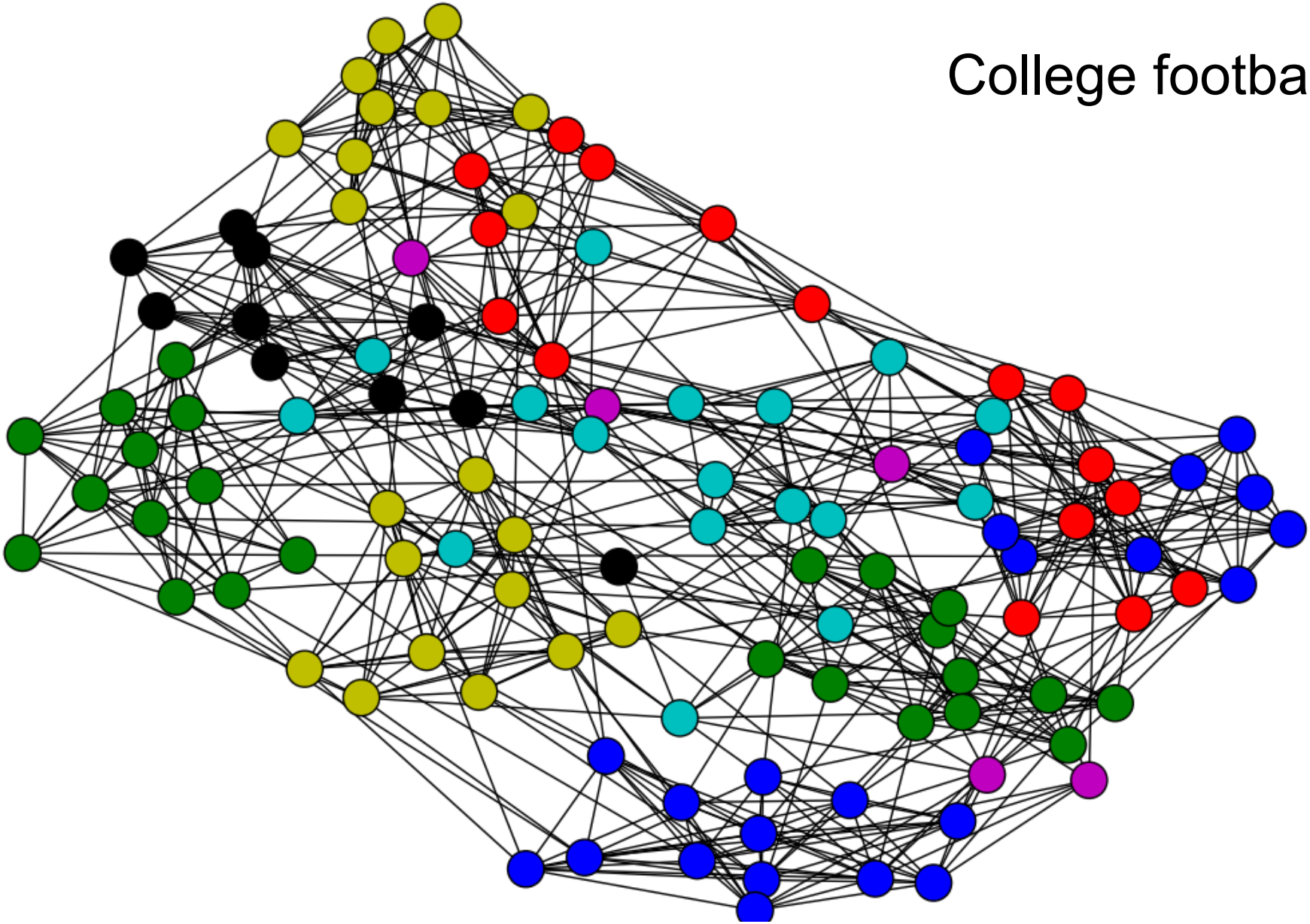


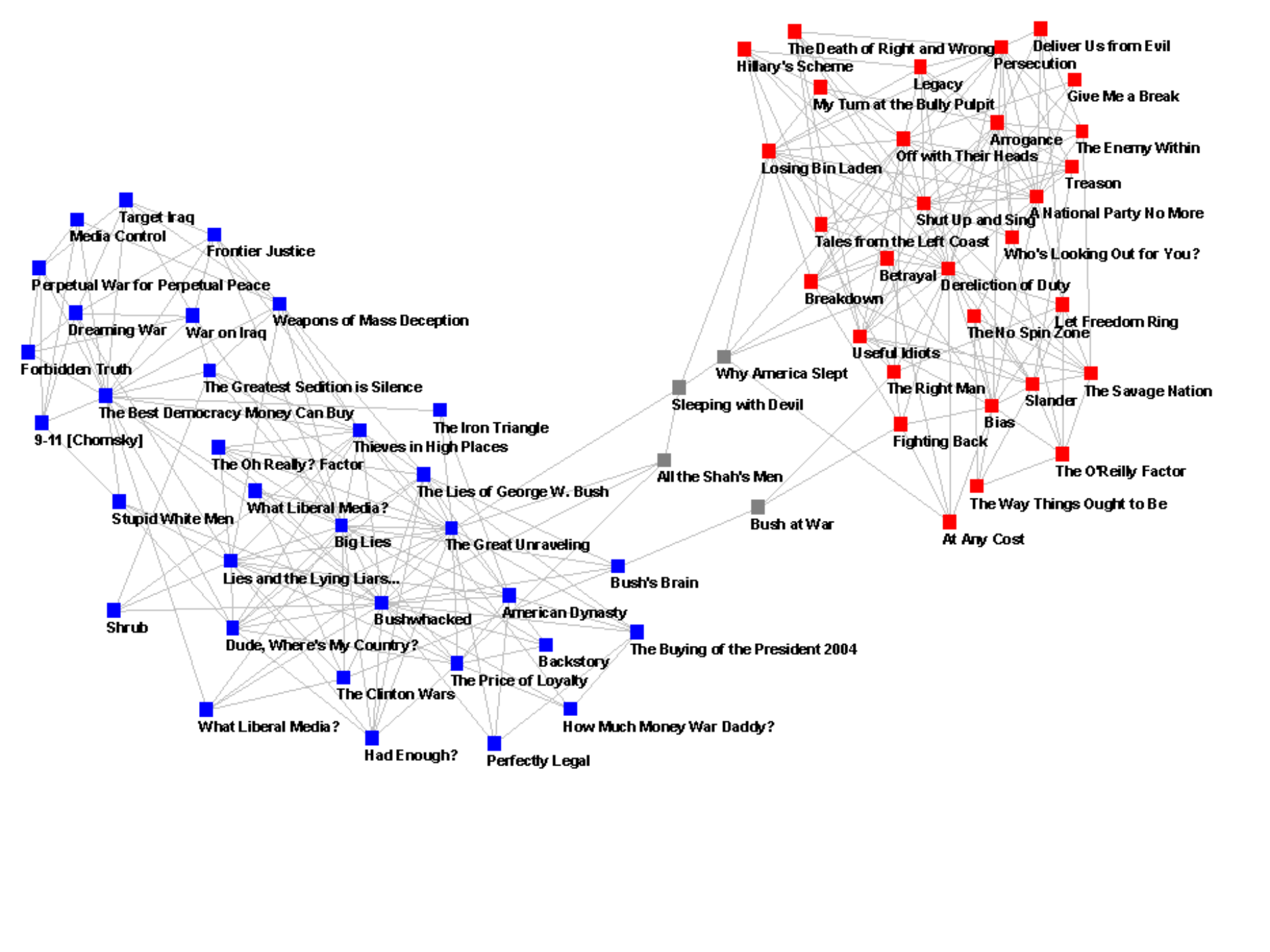
Karate club

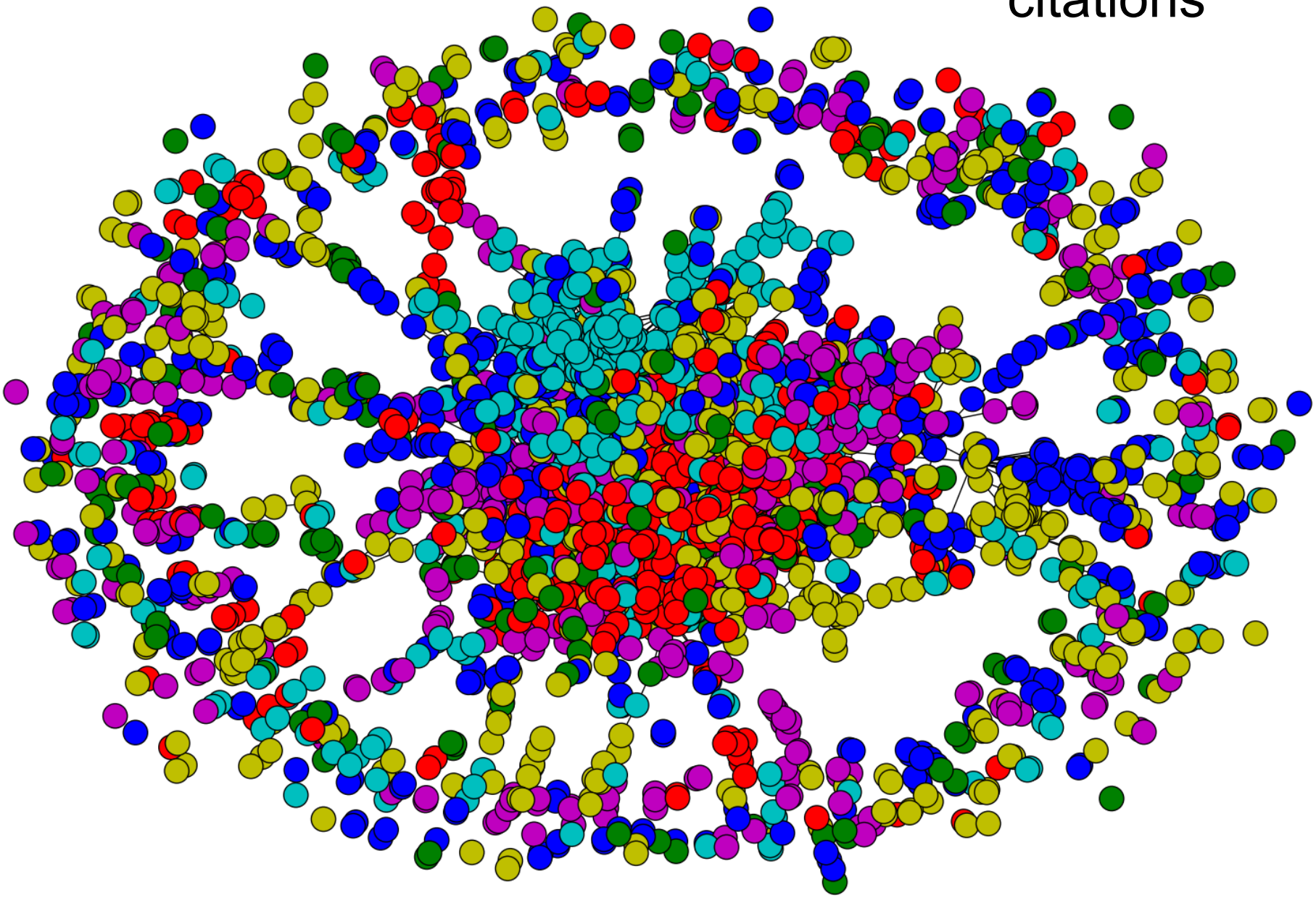

23
27
15
10
16
31
20
13
30
34
14
4
5
21
33
9
6
17
19
3
2
1
7
29
24
32
11
28
8
22
18
12
26
25

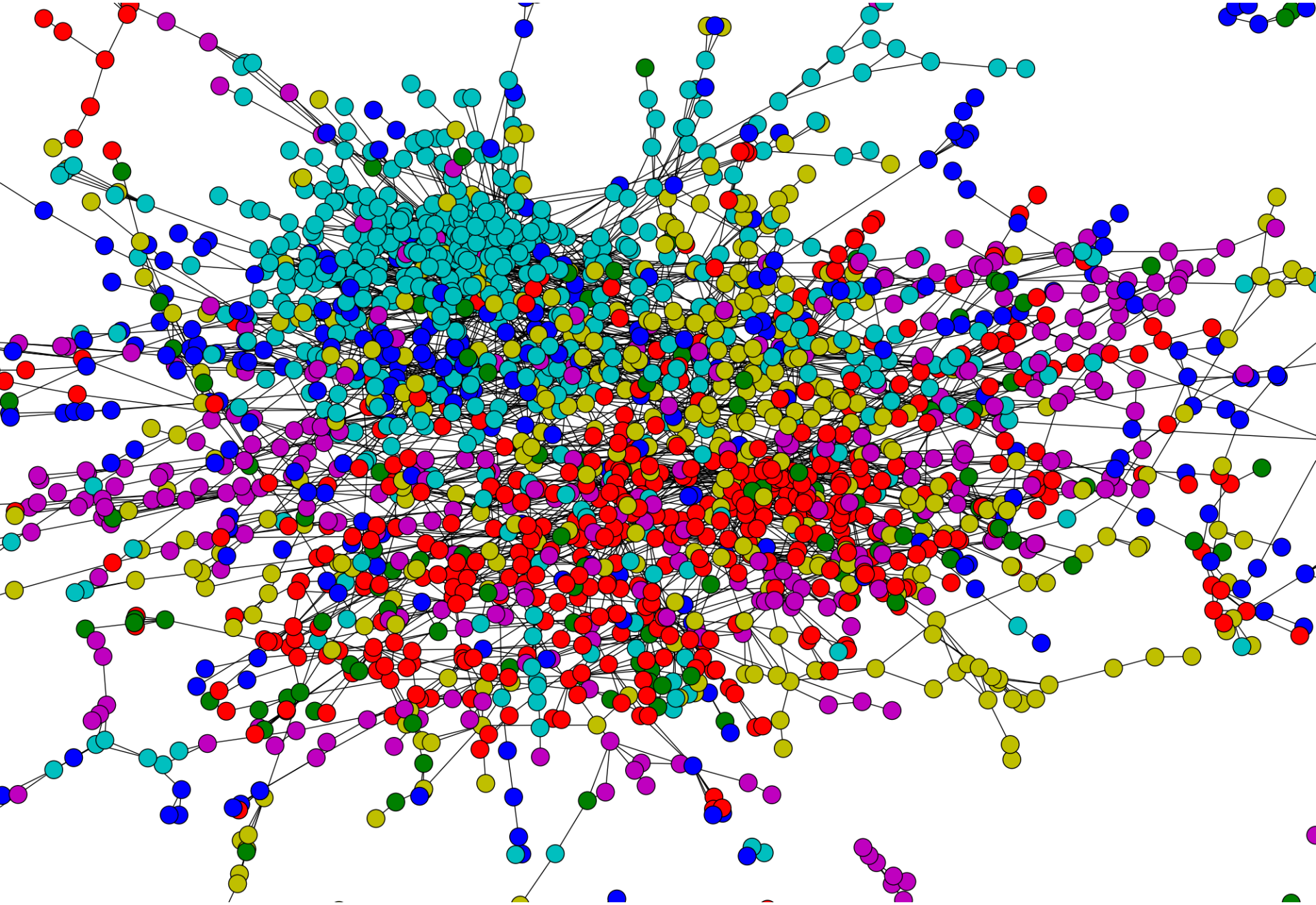
Schoolkids

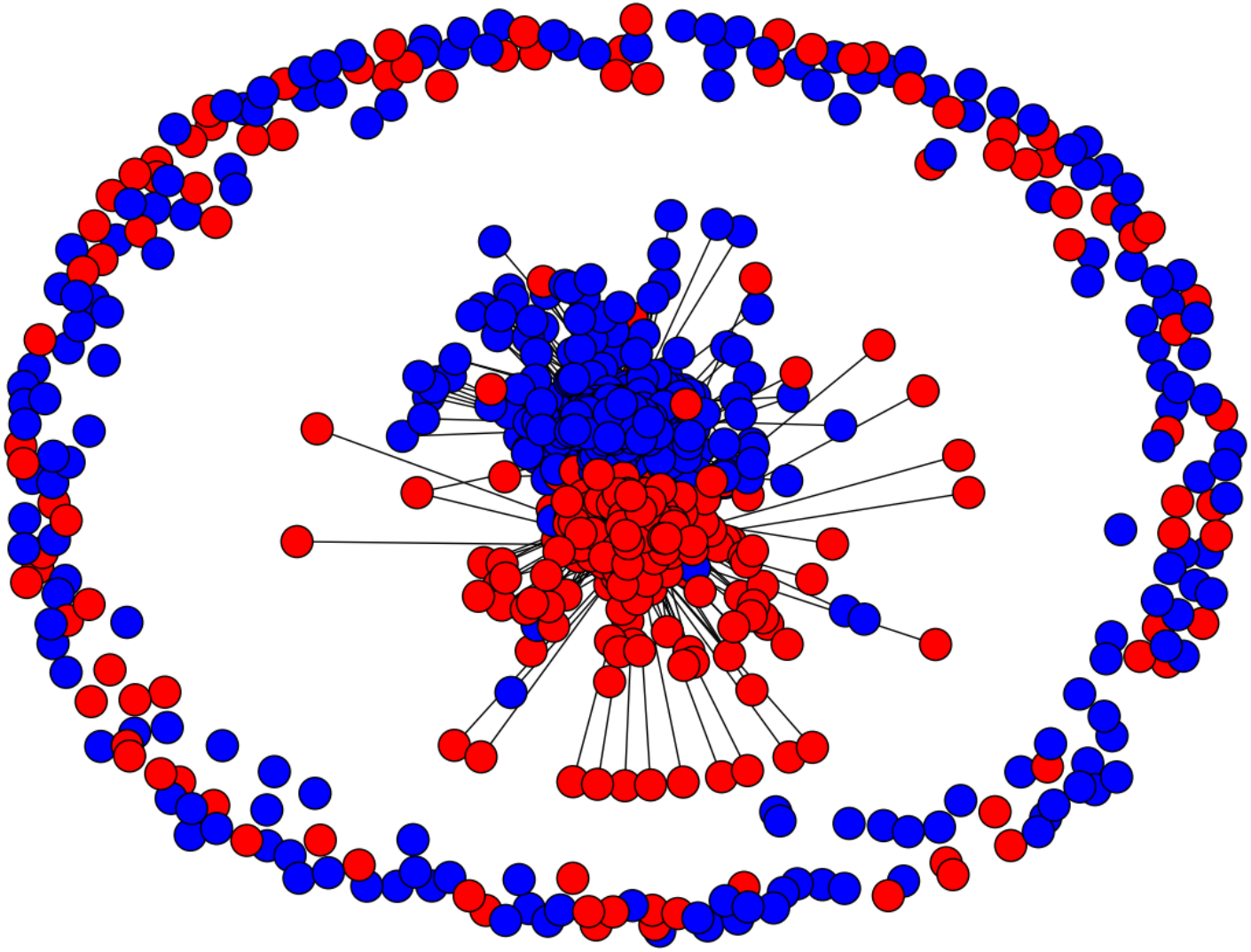


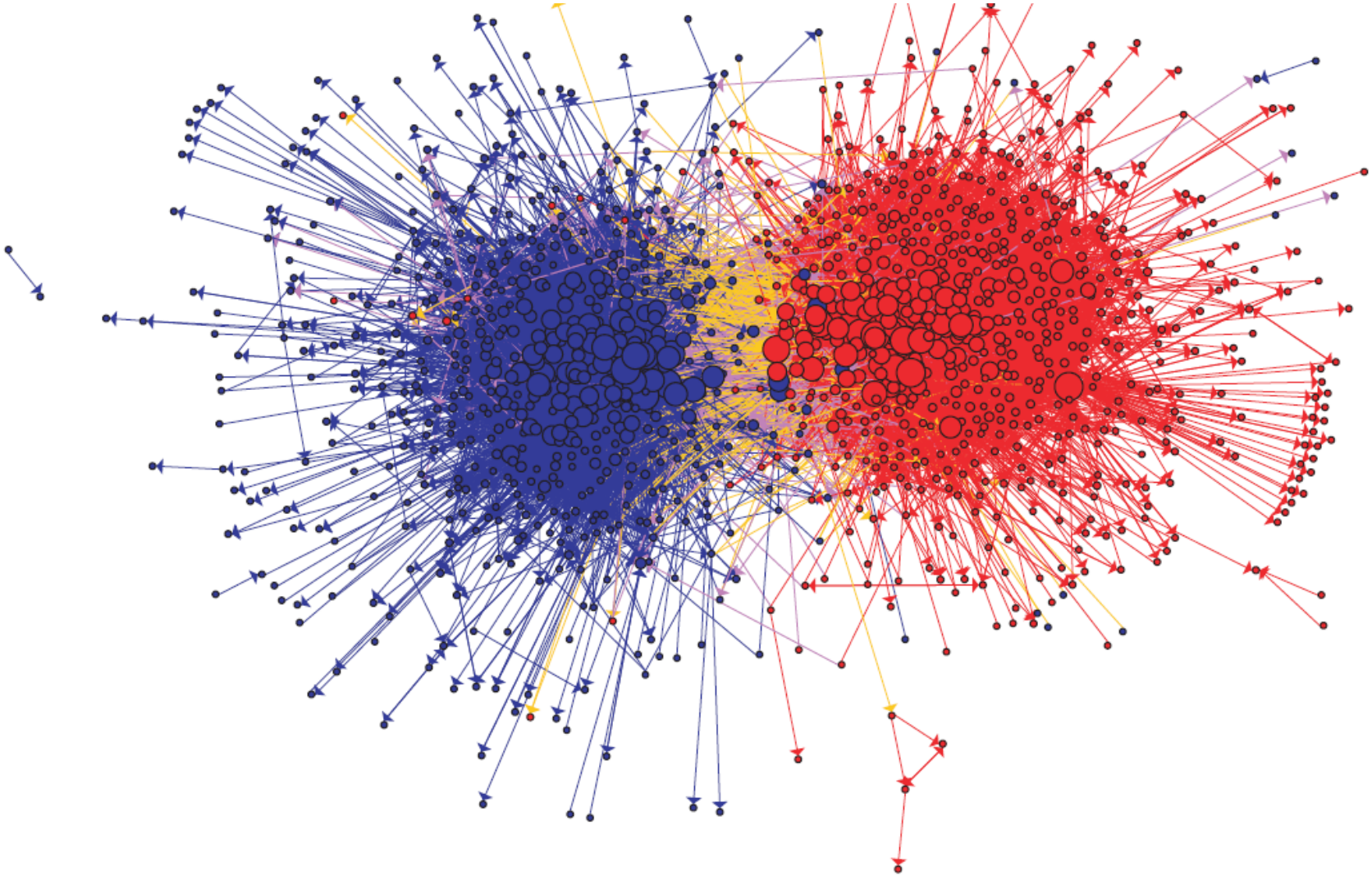

College football


The Death of Right and Wrong
Deliver Us from Evil
Hillary's Scheme
Persecution
Legacy
Give Me a Break
My Turn at the Bully Pulpit
Arrogance
The Enemy Within
Off with Their Heads
Losing Bin Laden
Treason
Shut Up and Sing
A National Party No More
Tales from the Left Coast
Who's Looking Out for You?
Betrayal
Dereliction of Duty
Breakdown
Let Freedom Ring
The No Spin Zone
Useful Idiots
Why America Slept
The Right Man
Slander
The Savage Nation
Sleeping with Devil
Bias
Fighting Back
The O'Reilly Factor
All the Shah's Men
The Way Things Ought to Be
Bush at War
At Any Cost
Target Iraq
Media Control
Frontier Justice
Perpetual War for Perpetual Peace
Weapons of Mass Deception
Dreaming War
War on Iraq
Forbidden Truth
The Greatest Sedition is Silence
The Best Democracy Money Can Buy
The Iron Triangle
9-11 [Chomsky]
Thieves in High Places
The Oh Really? Factor
The Lies of George W. Bush
What Liberal Media?
Stupid White Men
Big Lies
The Great Unraveling
Lies and the Lying Liars...
Bush's Brain
American Dynasty
Shrub
Bushwhacked
Dude, Where's My Country?
The Buying of the President 2004
Backstory
The Price of Loyalty
The Clinton Wars
What Liberal Media?
How Much Money War Daddy?
Had Enough?
Perfectly Legal


citations

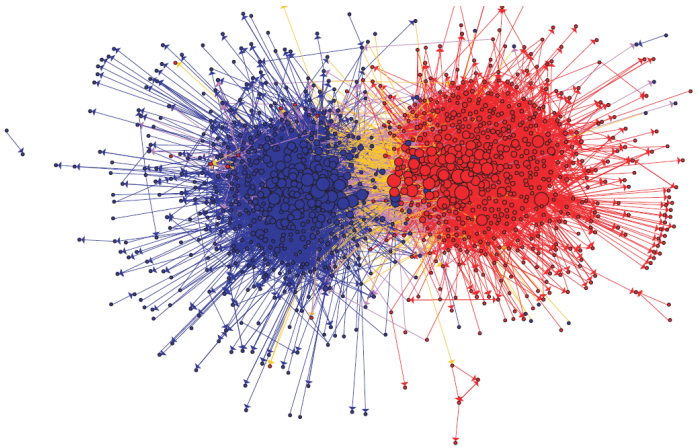






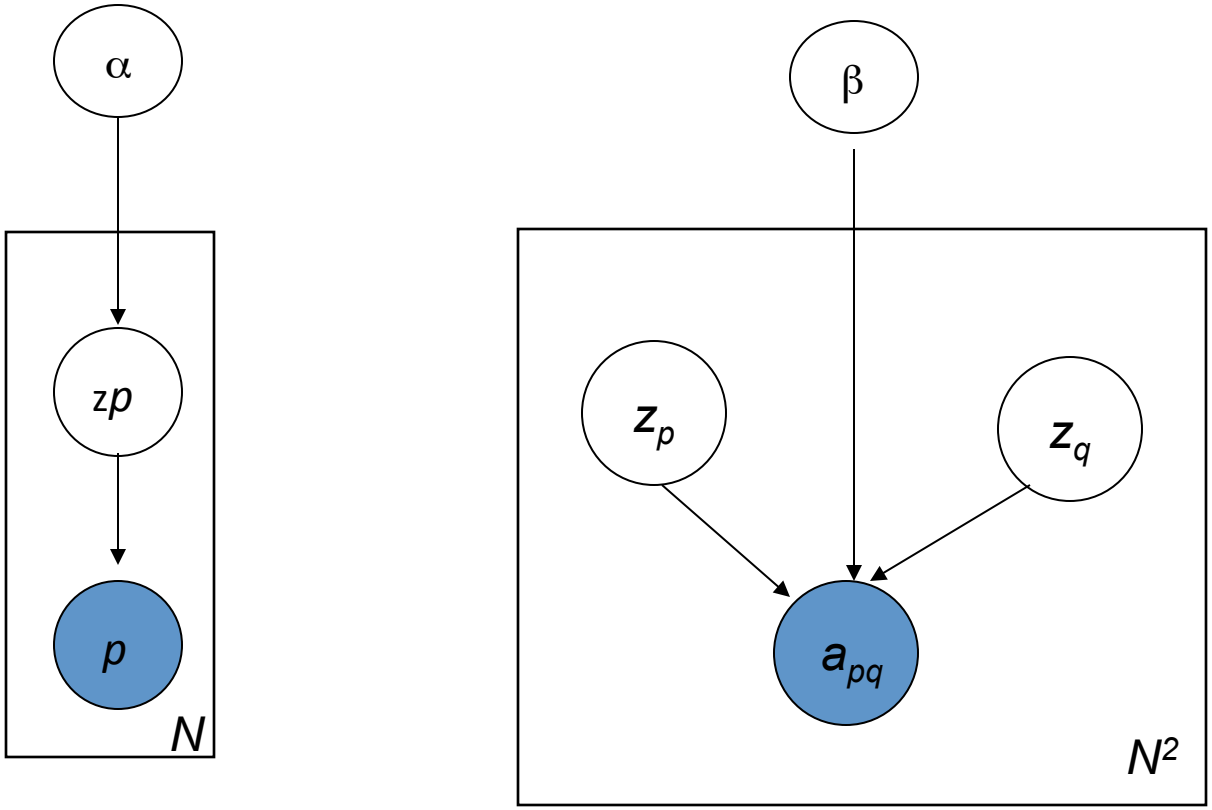
Stochastic Block Models

- "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 $i \neq j$



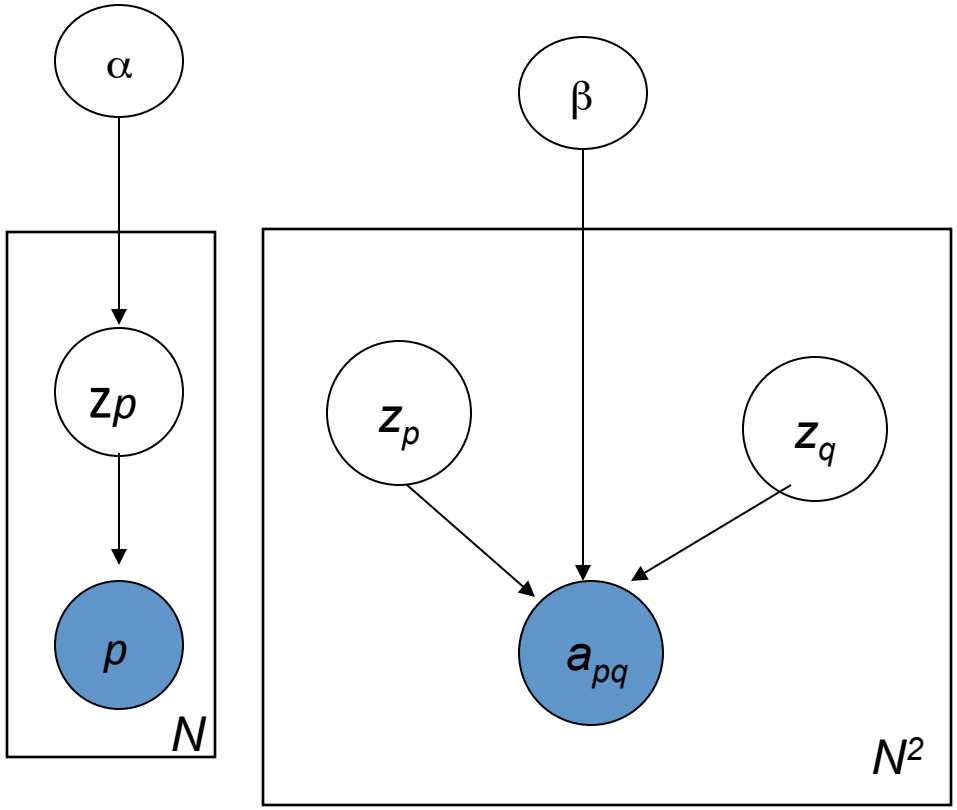
Stochastic Block models:

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



Stochastic Block models:

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

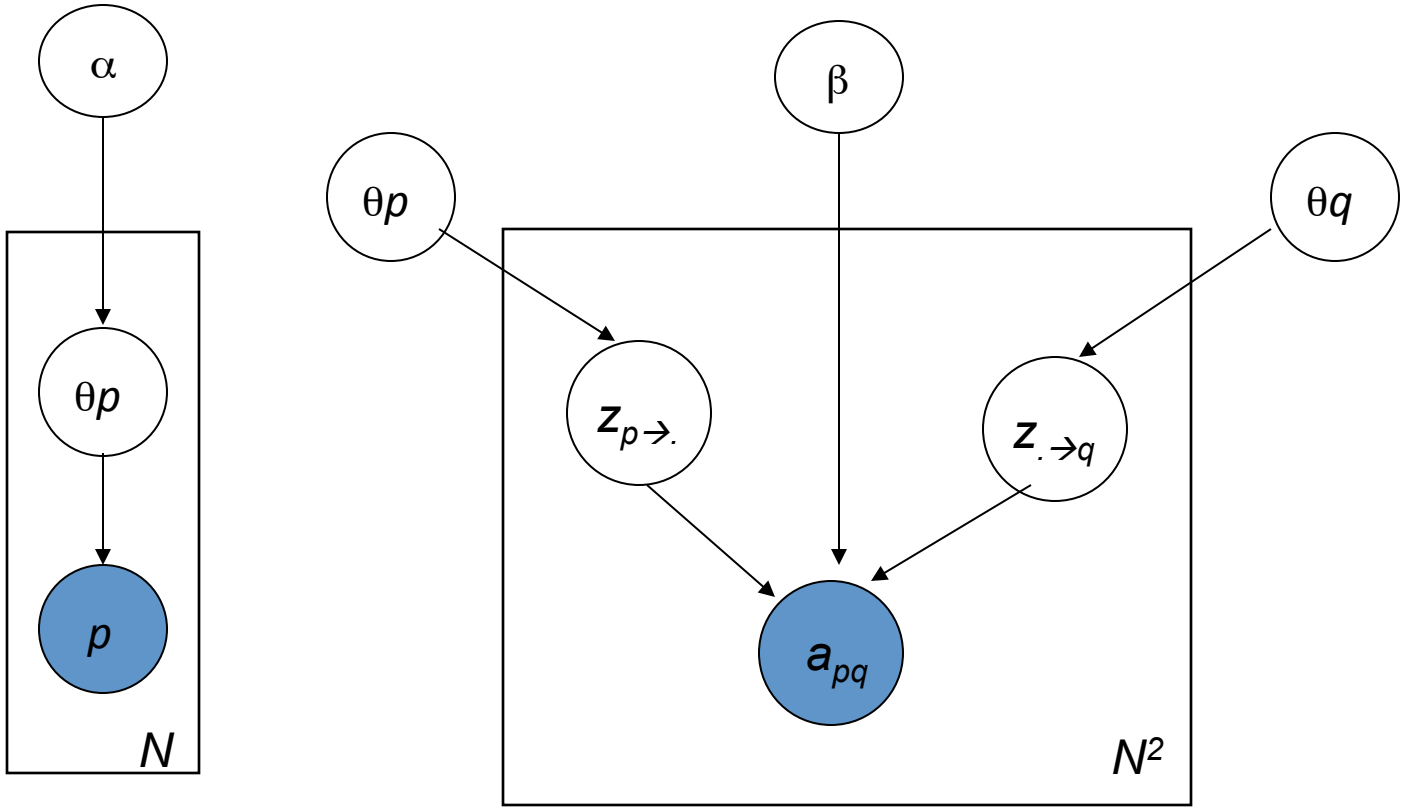


Gibbs sampling:

- Randomly initialize z_p for each node p.
- For t = 1…
  - For each node p
    - Compute z_p given other z's
    - Sample z_p

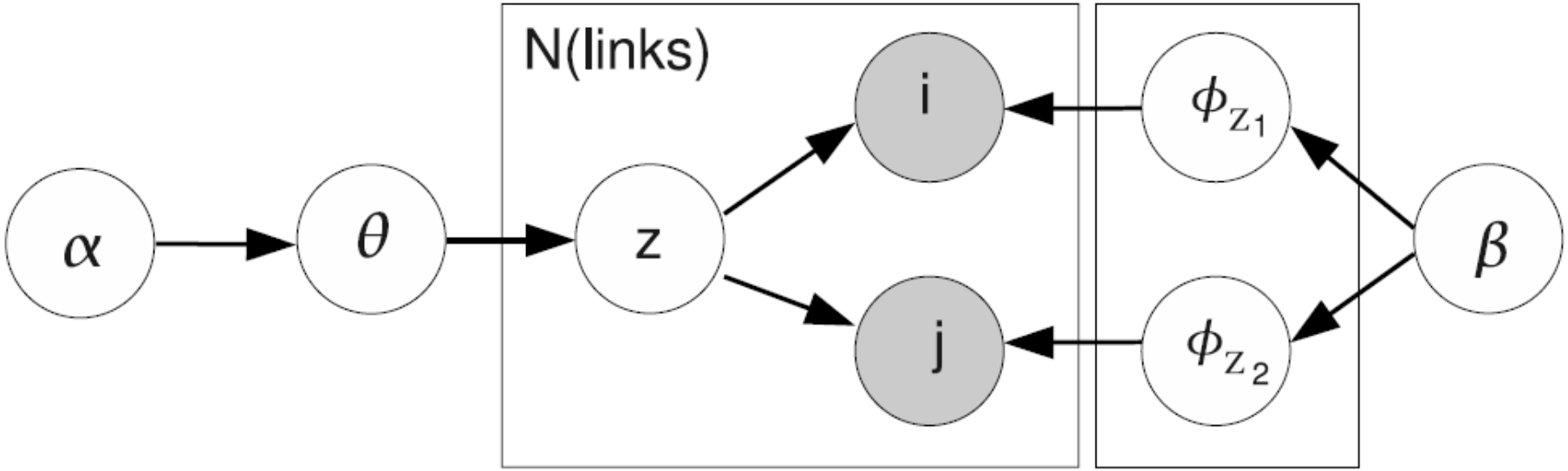
See: Snijders & Nowicki, 1997, Estimation and Prediction for Stochastic Blockmodels for Groups with Latent Graph Structure

Mixed Membership Stochastic Block models



Airoldi et al, JMLR 2008

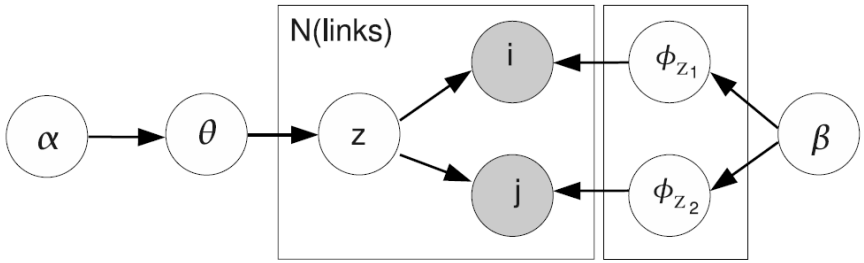
Another mixed membership block model



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

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

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

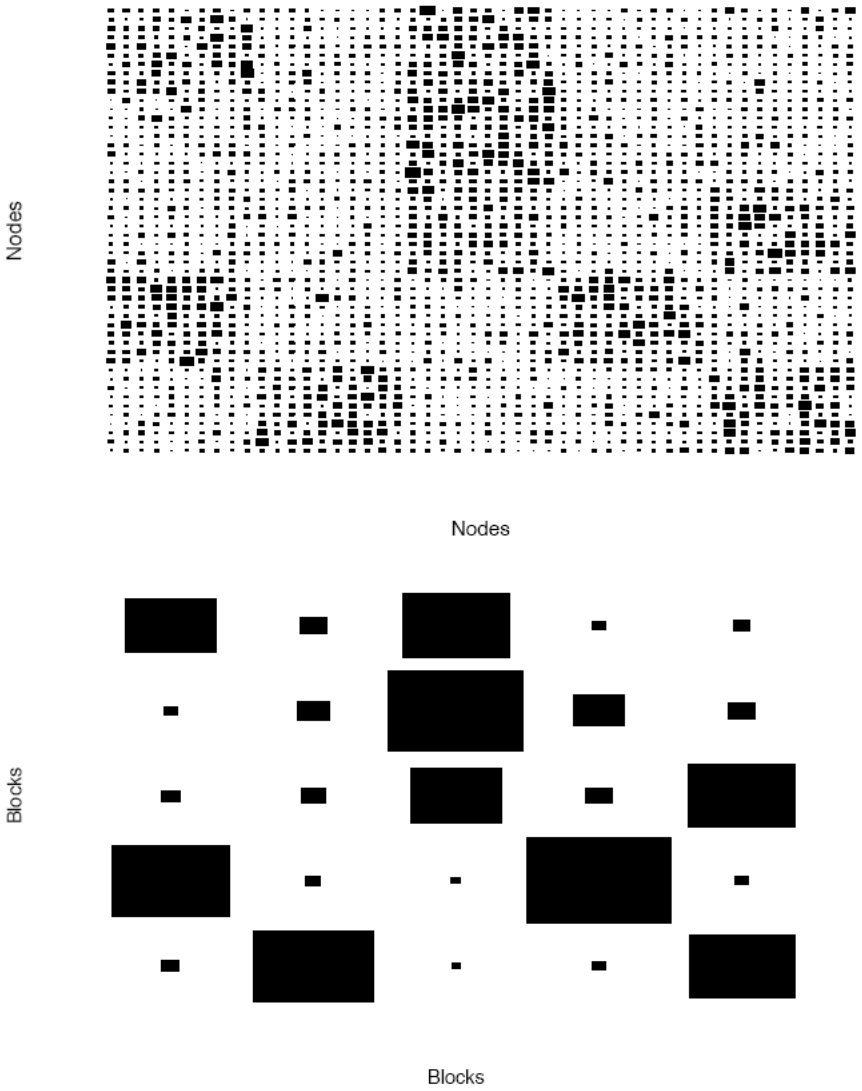
δ = indicator for diagonal

M = #nodes

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

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

Another mixed membership block model



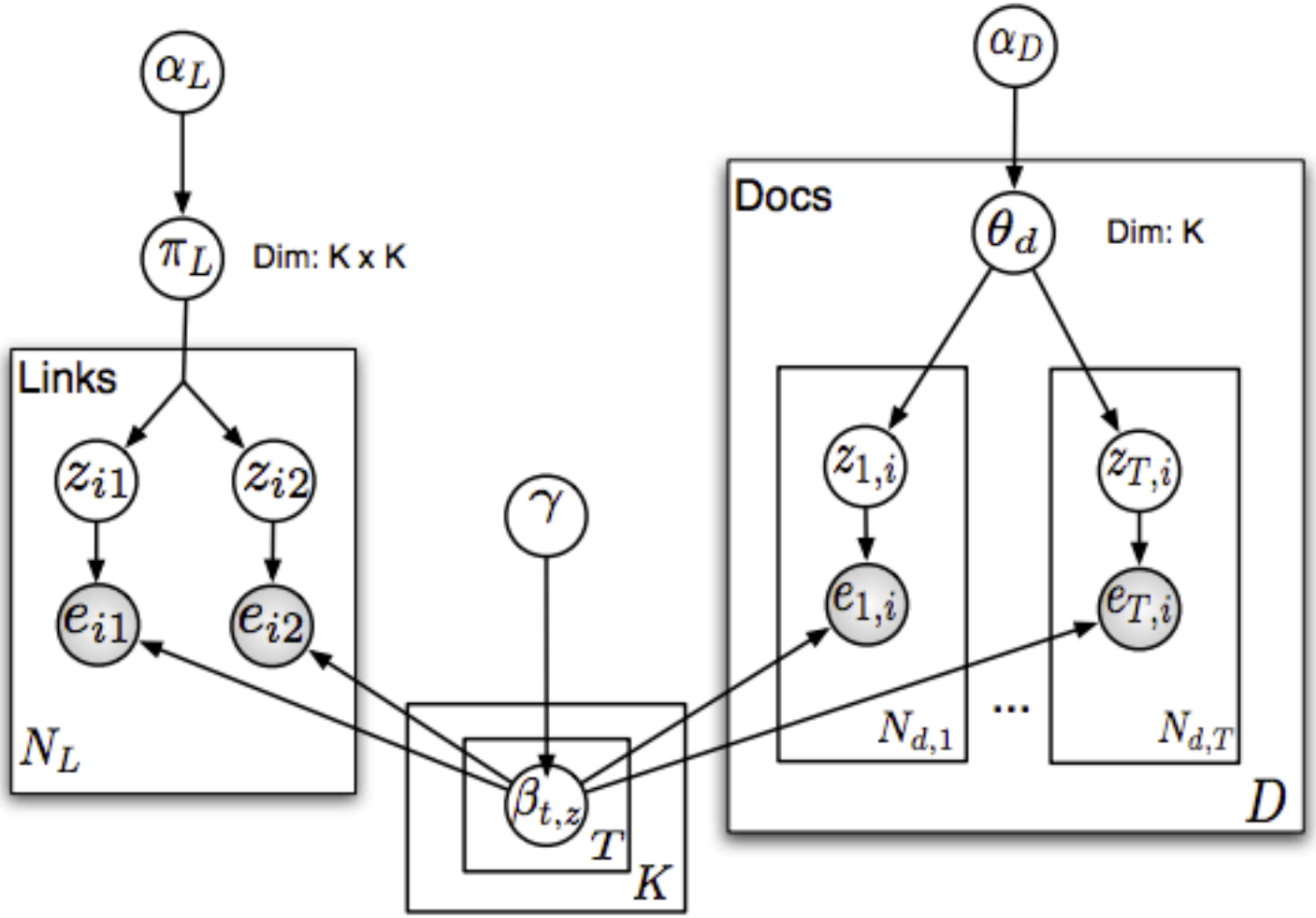
Sparse network model vs spectral clustering

(c) Best alignment: Social networks

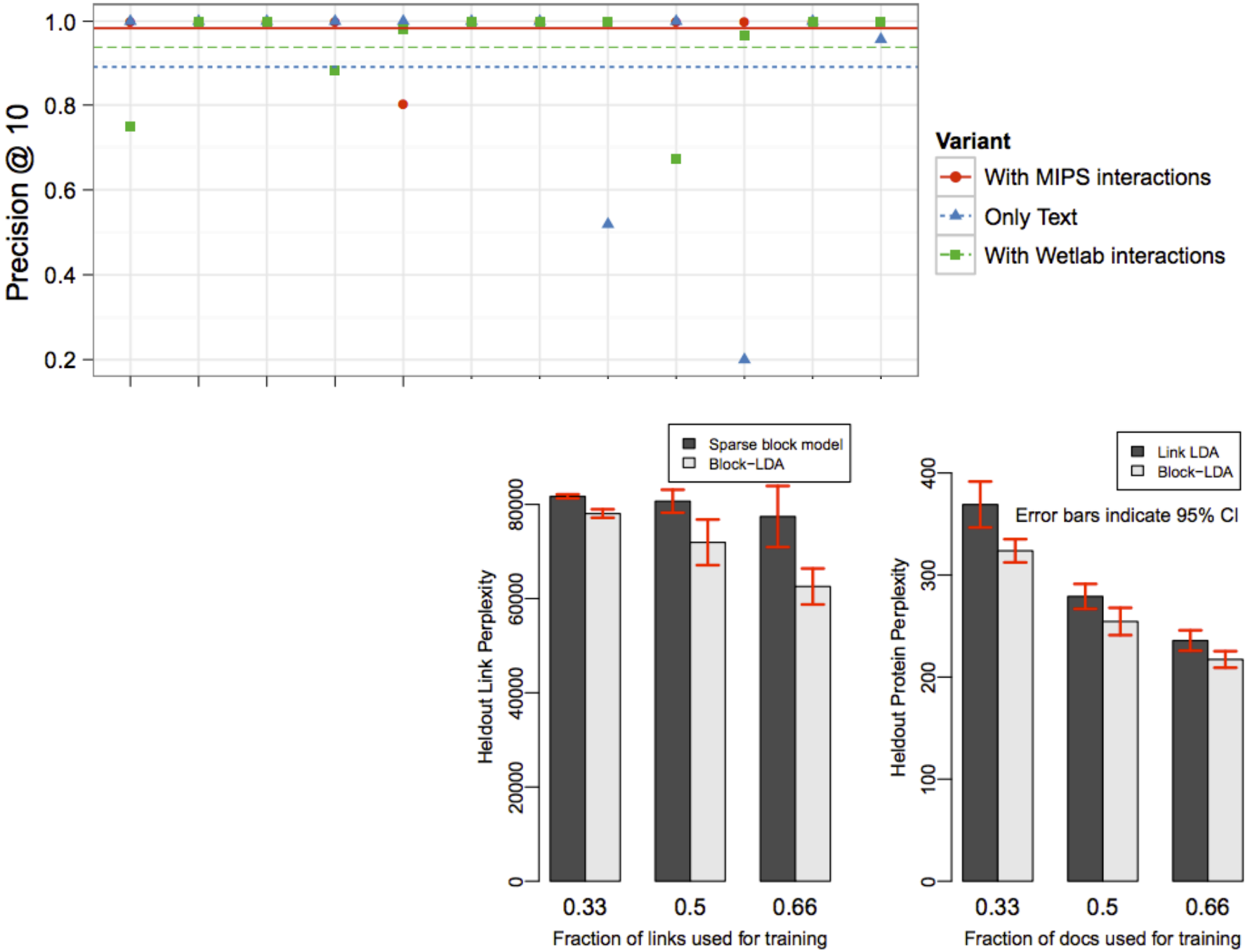
Dataset	PSK	NCut	NJW
Karate	**1.00**	0.95	0.95
Dolphin	0.90	**0.98**	**0.98**
UMBC	0.95	0.95	**0.96**
AG	**0.95**	0.52	0.51
MSP	**0.88**	0.63	0.64
Senate	0.98	**0.99**	**0.99**
PolBook	0.78	0.82	0.80
Football	**0.76**	0.72	0.67
MGEmail	0.28	0.59	0.56
CiteSeer	0.33	0.48	0.52
Cora	**0.47**	0.29	0.42
Average	0.75	0.72	0.73

Hybrid models

Balasubramanyan & Cohen, BioNLP 2012



Topic	Top Words & Proteins
Protein Structure & Interactions (Publications Only)	**Words:** protein structure binding residues domain structural beta complex atp proteins alpha interactions folding structures form terminal peptide helix model interaction bound domains molecular changes conformational **Proteins:** CYC1 SSA1 HSP82 SUP35 HSP104 HSC82 SSA2 YDJ1 URE2 KAR2 SSB1 SSA4 GCN4 SSA3 SSB2 PGK1 PDI1 SSC1 HSP60 STI1 SIS1 RNQ1 SEC61 SSE1 CCP1
DNA Repair (Using MIPS PPI)	**Words:**dna recombination repair replication strand single double cells mutations stranded induced base uv mutants mutation homologous virus telomere human type yeast activity telomerase mutant dna_polymerase **Proteins:** RAD52 RAD51 RAD50 MRE11 RAD1 RAD54 SGS1 MSH2 RAD6 YKU70 REV3 POL30 RAD3 XRS2 RAD18 RAD2 POL3 RAD27 YKU80 RAD9 RFA1 TLC1 TEL1 EST2 HO
Vesicular Transport (Using Wetlab PPI)	**Words:**membrane protein transport proteins atp golgi er atpase membranes plasma_membrane vesicles cells endoplasmic_reticulum complex fusion ca2 dependent translocation vacuolar intracellular yeast lipid channel hsp90 vesicle **Proteins:** SSA1 HSP82 KAR2 PMA1 HSC82 SEC18 SSA2 YDJ1 SEC61 PEP4 HSP104 SEC23 VAM3 IRE1 SEC4 SSA4 SEC1 PMR1 PEP12 VMA3 VPH1 SSB1 VMA1 SAR1 HAC1


Precision @ 10
1.0
0.8
0.6
0.4
0.2
Variant
With MIPS interactions
Only Text
With Wetlab interactions
Sparse block model
Block−LDA
Heldout Link Perplexity
80000
60000
40000
20000
0
0.33
0.5
0.66
Fraction of links used for training
Link LDA
Block−LDA
Error bars indicate 95% CI
Heldout Protein Perplexity
400
300
200
100
0
0.33
0.5
0.66
Fraction of docs used for training