Discrete Principal Components Analysis

Wray Buntine

(Work with help from Aleks Jakulin, Sami Perttu

and lots of other folks at COSCO)

Helsinki Inst. for Information Technology (HIIT)

Subspace, Latent Structure and Feature Selection techniques:
Statistical and Optimisation perspectives Workshop,
Bohinj, Slovenia
23nd February, 2005

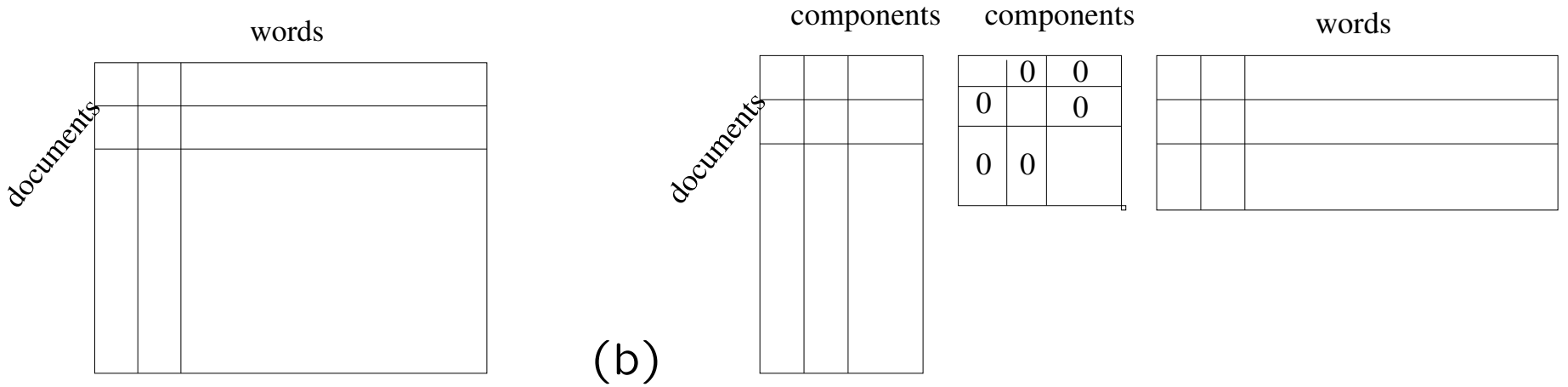
Discrete Components Analysis

- Alphabet soup of methods more or less the same up to some distributional choices, normalisation* and statistical religion: GOM, PLSA, NMF, LDA, GaP, MPCA, ...
- Corresponds to multifaceted clustering on discrete data.
- Is model-based Independent Components Analysis (ICA) in discrete domain.
- Is a multinomial analogue to Gaussian interpretations of LSI/PCA.
- Non-negative matrix factorisation is a good analogy.
- Earliest precursor in text is bi-clustering of Pereira, Tishby and Lee, 1993.
- Biggest users are in genotype analysis using Pritchard *et al.* 2000 Gibbs implementation.

* Note normalising gammas yields a Dirichlet, and normalising Poissons yields a multinomial.

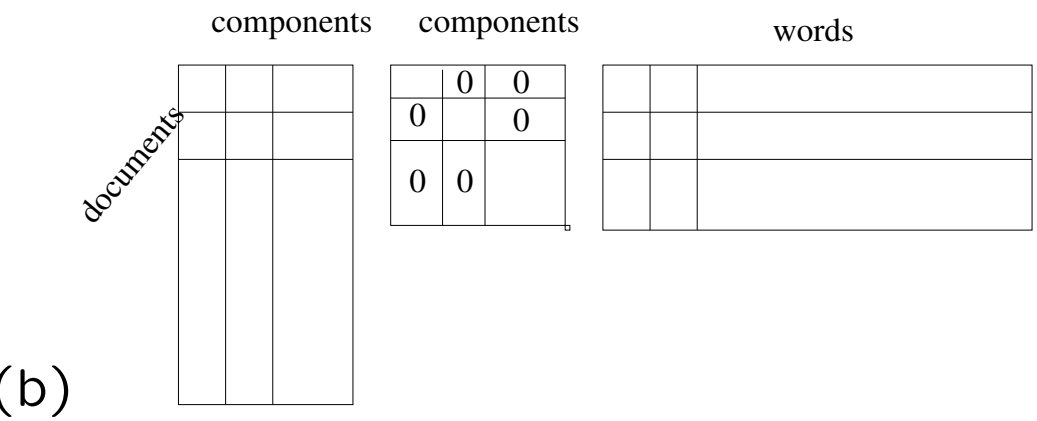
Matrix factorisation

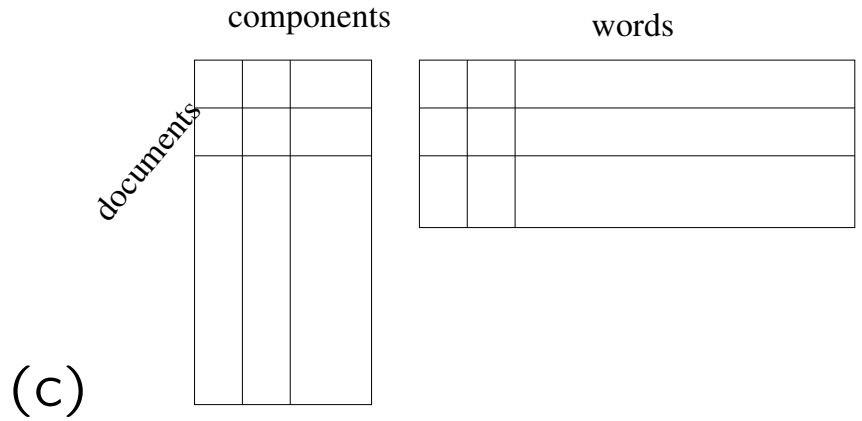
- Standard SVD, decomposes matrix $D = U^T \Lambda V$.
- U and V have rows mutually orthogonal and normalised using L_2 norm.
- Λ is diagonal, and is scaling.
- Principal components analysis (PCA) zeros the smaller values in Λ (as "noise"), making U and V non-square.
- Thus (b) approximates (a) in a least squares sense.



Matrix factorisation: statistical view

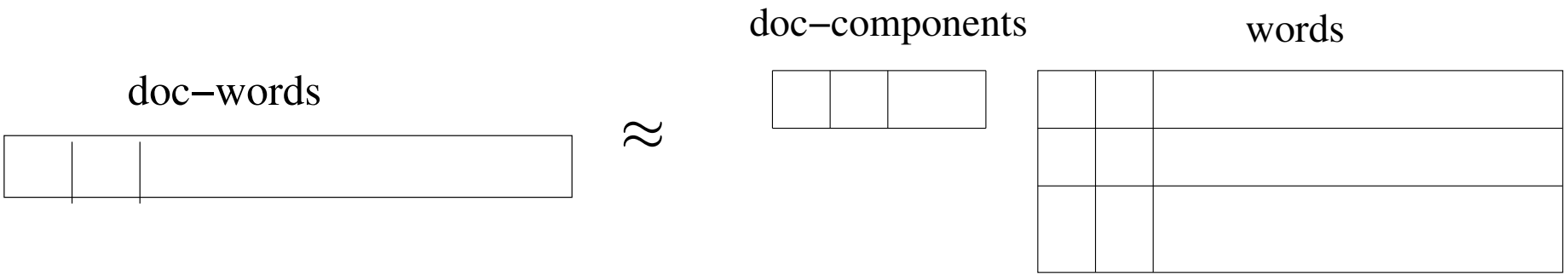
- Zero smaller eigenvalues in Λ as noise. U and V not square but rows still normalised (in L_2) and orthogonal.
- Fold $U^T\Lambda$ in together. It is `documents` $\times$ `components`. This is the document specific part. These are *hidden variables*: the projection in the lower dimensional space.
- Leave V alone. It is `components` $\times$ `words`. This is the intrinsic basis for "typical documents". Is lower dimensional space on which documents are projected.
- Thus (b) becomes (c).





Matrix factorisation: single document view

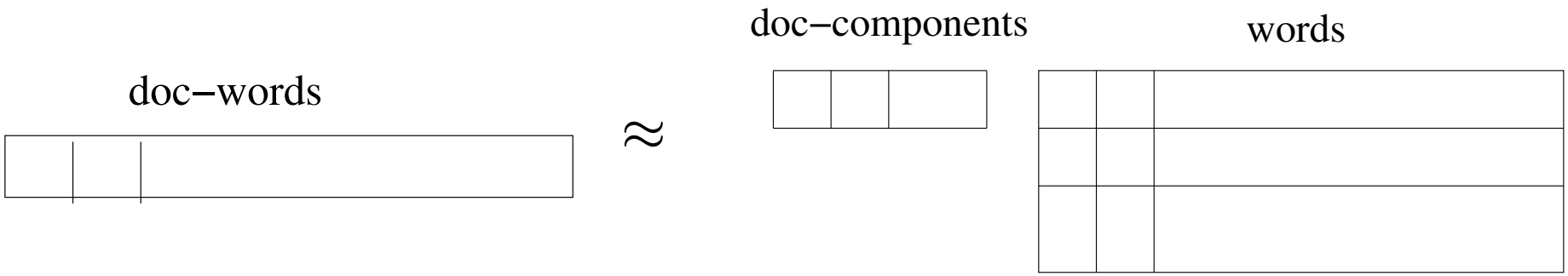
- One row of the document matrix represents *hidden variables* for a single document.
- The entries might, for instance, be independently distributed.
- Thus a single document is approximated as a dot product of the bases.



Problem: *the bases are of mixed sign, so what do they represent?*

Idea: *lets do it right statistically!*

Matrix factorisation: single document model



line of (a) line of (c)

- `doc-words` is the data vector $\vec{w}$. Given.
- `doc-components` is the hidden variable (a vector) $\vec{l}$. Use $l_k \sim \Gamma(\alpha_k, \beta_k)$. l_k is interpreted as number of words in component k.
- The `components` $\times$ `words` matrix is the *model matrix*, $\mathbf{\Theta}$. We normalise it along rows. Each row is the word rates/probabilities for one component.
- The relationship between LHS (data) and RHS (estimate from hidden data) can be approximating by discrete distributions: Poissons, multinomials, e.g., Kullback-Leibler distance.

$$\langle \vec{w} \rangle = \vec{l} \cdot \mathbf{\Theta}$$

Example problem: Wikipedia

- We build a $K = 100$ component model of 290,000 web pages from the English-language Wikipedia* dated July 2004.
- Discrete PCA is run on bags of lemmatised words.
- Word lemmas grouped into noun, verb, adjective, adverb and other word classes discarded.
- Thus data vector $\vec{w}$ is a sparse vector of dimension about $300,000$, partitioned into four groups, with about $240,000$ nouns. But with only about 20-1000 non-zero entries.

*`http://www.wikipedia.org`

Example component: "Mythology"

Distinctive phrases: have a high Bayes factor with the component.

> God; name; spirit; goddess; Greek mythology; Holy Spirit; Greek word; Polynesian mythology; Golden Age; evil spirits;

High ranked document titles: use the component as the topic in topic-specific page rank (Richardson and Domingos, 2001), to get page rankings based on topical content and links.

> Greek mythology; Roman mythology; Celtic mythology; Norse mythology; Underworld; Aztec mythology;

High probability lemmas: Table on the next page gives one row from Θ for a component. It is a sparse vector of length about $300,000$ so just the high frequency terms are listed. Note we have 4 multinomials packed into one vector.

NOUNS					
mythology	0.03337	God	0.02048	name	0.014747
goddess	0.012911	spirit	0.012639	legend	0.0087992
myth	0.0070882	demons	0.006807	Sun	0.0060099
Temple	0.0054717	deity	0.0054247	Bull	0.0051629
Dragon	0.0051379	Maya	0.0051243	King	0.00512
Sea	0.0049453	Norse	0.0044707	horse	0.0044592
symbol	0.0042196	animals	0.0040112	fire	0.0039879
hero	0.0038755	Romans	0.0038696	Apollo	0.0037588
Lion	0.0036306	Earth	0.0035993	giant	0.0035076
VERBS					
called	0.034078	said	0.031081	See	0.029521
given	0.0269	associated	0.024591	According	0.021724
represented	0.020964	known	0.018896	could	0.017499
made	0.016952	depicted	0.01524	appeared	0.014662
ADJECTIVES					
Greek	0.091163	ancient	0.055393	great	0.02853
Egyptian	0.028071	Roman	0.025783	sacred	0.020446
ADVERBS					
sometimes	0.058205	often	0.048815	so	0.046594

Gamma-Poisson model*

- I is documents, J is words/features, K is components.
- Hidden variable (a vector) $\vec{l}$. For each component $k = 0, ..., K-1$

$$l_k \sim \Gamma(\alpha_k, \beta_k) \ .$$

- The $K \times J$ *model matrix* is $\mathbf{\Theta}$. Entries $\theta_{j,k}$. We normalise it along rows.
- For each feature/word j

$$w_j \sim \text{Poisson}\left(\sum_k l_k \theta_{j,k}\right)$$

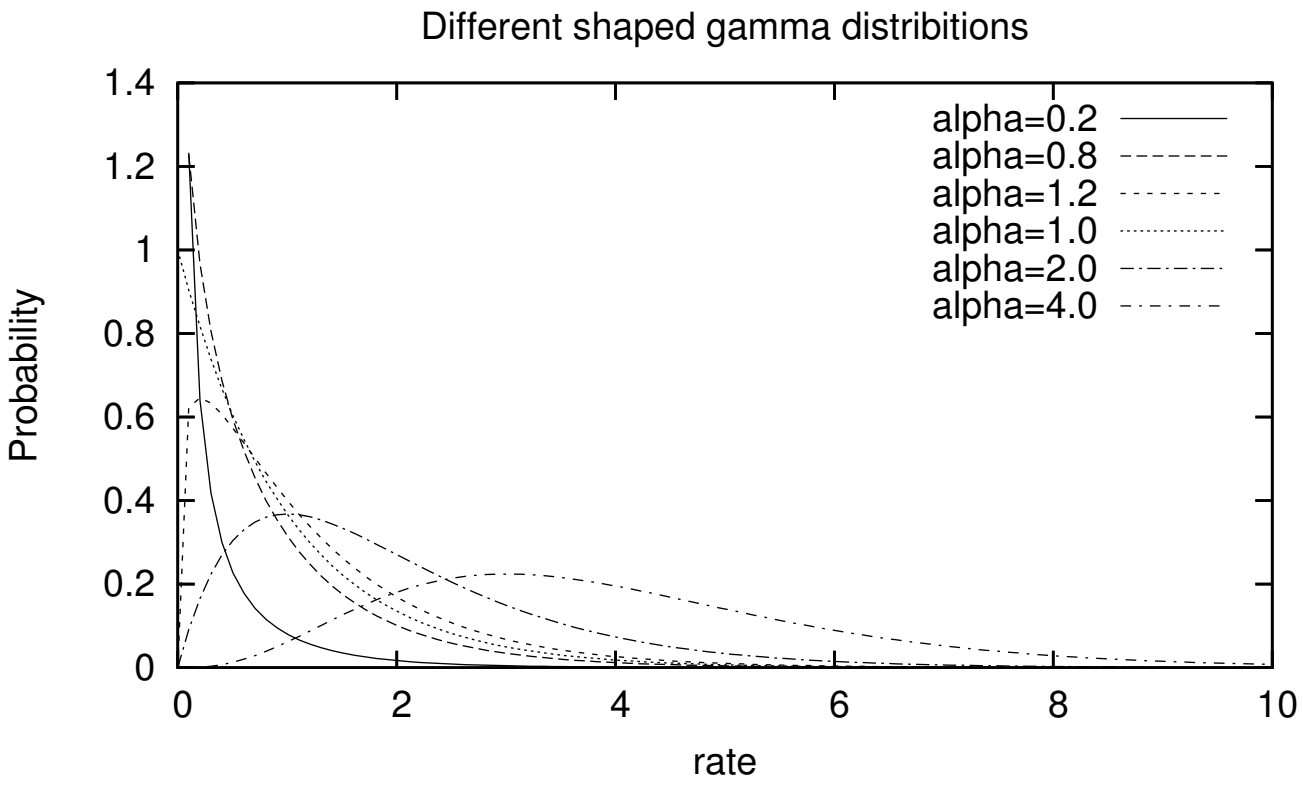
* Introduced by Canny, SIGIR 2004. Version of non-negative matrix factorisation (NMF) of Lee and Seung (1999). Correspondence with ICA noted Canny 2004.

Gamma components

- components

  $l_k \sim \Gamma(\alpha_k, \beta_k)$.

- For $\alpha_k > 4$ approaches Gaussian.



Dirichlet-multinomial model*

Start with the Gamma-Poisson. Assume the β_k are constant, given by β.

- Total of $\vec{l}$ is $\sum_k l_k \sim \Gamma\left(\sum_k \alpha_k, \beta\right)$. Hidden variable (a vector) $\vec{m} = \vec{l}/\left(\sum_k l_k\right)$ distributed $\vec{m} \sim \text{Dirichlet}_K(\vec{\alpha})$.
- The word total $L = \sum_j w_j$ is now

$$L \sim \text{Poisson}\left(\sum_k l_k\right)$$

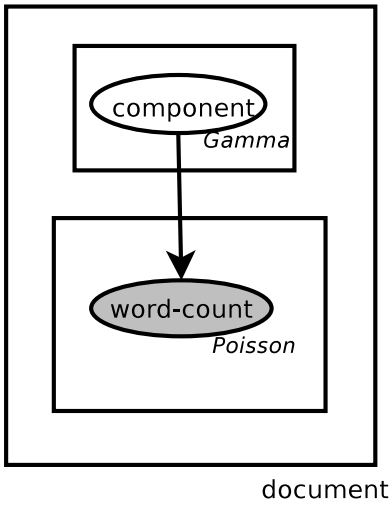
- The vector of word counts now jointly multinomial

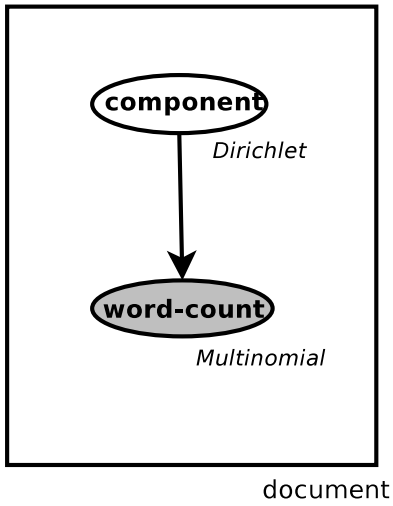
$$\vec{w} \sim \text{multinomial}\left(\left\{\sum_k m_k \theta_{j,k} : j\right\}, L\right)$$

* Version of PLSI introduced by Hofmann 1999, and admixture modelling of Pritchard *et al.* 2000. Later called LDA by Blei and Jordan, 2002 and multinomial PCA by Buntine 2002. Correspondence with NMF noted Buntine 2002. Correspondence with ICA noted Buntine and Jakulin 2004.

Discrete PCA Versions

- On left is GaP model from Canny 2004
- Normalise to get the right, the multinomial PCA model of Buntine 2002 (assumes β of gammas is constant).





Discrete PCA general model

- Components l_k generated by independent gammas $\Gamma(\alpha_k, \beta_K)$.
- Data $\vec{w}$ partitioned into Poisson and/or multinomial parts.
- Mean of Poisson part for word j is dot of $\vec{l}$ and j-th column of $\mathbf{\Theta}$.
- Mean of each entry in multinomial part, a word j, is dot of $\vec{l}/\sum_k l_k$ and j-th column of $\mathbf{\Theta}$.
- Variations: *gamma-Poisson* (no multinomials), *Dirichlet-discrete* (β_k constant), else *Gamma-discrete*.

Likelihood with Components

Full likelihood for a document with hidden variables $\vec{l}$ is

$$p(\vec{w}, \vec{l} \,|\, \text{discrete PCA}, K, \vec{\alpha}, \vec{\beta}, \mathbf{\Theta})$$

Formula		Source
$\prod_k \frac{\beta_k^{\alpha_k}}{\Gamma(\alpha_k)} l_k^{\alpha_k - 1} e^{-\beta_k l_k}$	...	K component gammas.
$e^{-\delta_P (\sum_k l_k)} \prod_{j \in B_1} \frac{1}{w_j!} \left(\sum_k l_k \theta_{j,k} \right)^{w_j}$	...	Several Poissons for some words j in B_1.
$\prod_{g \in 2,\ldots,G}$	...	$G-1$ multinomials.
$\frac{L_g!}{(\sum_k l_k)^{L_g}} \prod_{j \in B_g} \frac{1}{w_j!} \left(\sum_k l_k \theta_{j,k} \right)^{w_j}$	...	Each multinomial for some sets of words j in B_g.

Discrete PCA hidden variable model

- Components $\vec{l}$ generated by independent gammas $\Gamma(\alpha_k, \beta_K)$.

- Introduce hidden variables, $v_{j,k}$ that are the number of words from the w_j words falling in component k. So $\sum_k v_{j,k} = w_j$. This means *every word is now tagged with its own component*.

- For each feature/word j and component k

$$v_{j,k} \sim \text{Poisson}\left(l_k \theta_{j,k}\right)$$

- However, since the total $\sum_k v_{j,k} = w_j$ must be obeyed when the data is given, this distribution should be replaced by the conditional

$$\{v_{j,k} : k = 1..K\} \sim \text{Multinomial}\left(\left\{\frac{l_k \theta_{j,k}}{\sum_k l_k \theta_{j,k}} : k = 1, \ldots, K\right\}, w_j\right) .$$

Likelihood with Components and Word Tagging

Full likelihood for a document with hidden variables $\vec{l}$ and $\mathbf{v}$ is

$$p(\mathbf{v}, \vec{w}, \vec{l} \,|\, \text{discrete PCA}, K, \vec{\alpha}, \vec{\beta}, \mathbf{\Theta})$$

Formula		Source
$\prod_k \frac{\beta_k^{\alpha_k}}{\Gamma(\alpha_k)} l_k^{\alpha_k - 1} e^{-\beta_k l_k}$	...	K component gammas.
$e^{-\delta_P \sum_k l_k} \prod_{j \in B_1} \prod_k \frac{(l_k \theta_{j,k})^{v_{j,k}}}{v_{j,k}!}$	...	Several Poissons for some words j in B_1.
$\prod_{g \in 2,\ldots,G}$	...	$G-1$ multinomials.
$\frac{L_g!}{(\sum_k l_k)^{L_g}} \prod_{j \in B_g} \prod_k \frac{(l_k \theta_{j,k})^{v_{j,k}}}{v_{j,k}!}$	...	Each multinomial for some sets of words j in B_g.

Discrete PCA interpretations for bag-of-words

For a given document:

- Components l_k are count of words in component k.
- Normalised to m_k, are proportion of words in component k. Hence called *multifaceted clustering*.
- Hidden variables $v_{j,k}$ means tagging each word by a particular component.
- Compare with regular clustering: one hidden topic per document, *versus* one hidden topic per word in the document.
- Word probabilities given by $\sum_k m_k \theta_{j,k}$, hence called *admixtures*.

Discrete PCA Text Applications

- feature discovery for text classification (works OK, SVMs better!)
- unsupervised synonym discovery;
- language models for text (unsupervised models on relations such as verb-subject) for use in a semantic parser
- language models for information retrieval (unsupervised models on bag-of-words); and
- for ontology discovery, hierarchical versions will be needed.

Discrete PCA Algorithm Issues

- The $l_k^{v_{j,k}}$ term in full likelihood means EM algorithm cannot be used to maximise $p(\vec{w}_1, \ldots, \vec{w}_I \,|\, \text{discrete PCA}, K, \vec{\alpha}, \vec{\beta}, \mathbf{\Theta})$.

  NB. conditional distributions for $\mathbf{v}$ and $\vec{l}$ not independent.

- However EM can apply to maximise $p(\vec{w}_1, \ldots, \vec{w}_I, \vec{l}_1, \ldots, \vec{l}_I \,|\, \text{discrete PCA}, K, \vec{\alpha}, \vec{\beta}, \mathbf{\Theta})$.

  NB. conditional distributions for $\mathbf{v}$ well behaved.

- Gibbs applies directly, but careful, see later.

  NB. all conditional distributions for $\mathbf{v}$ and $\vec{l}$ well behaved.

- Mean field applies too.

Mean Field

At each stop, like to maximize likelihood (or posterior probability)

$$\sum_i p(\vec{w}_{(i)} \,|\, \text{discrete PCA}, K, \vec{\alpha}, \vec{\beta}, \mathbf{\Theta})$$

Functional form leads to computational complexity. Thus, for appropriate posterior approximation $q(\mathbf{v}_{(i)}, \vec{l}_{(i)})$, maximize expected log. likelihood (or posterior probability)

$$\sum_i E_{q(\mathbf{v}_{(i)}, \vec{l}_{(i)})} \left\{ p(\vec{w}_{(i)} \,|\, \mathbf{v}_{(i)}, \vec{l}_{(i)}, \text{discrete PCA}, K, \vec{\alpha}, \vec{\beta}, \mathbf{\Theta}) \right\}$$

Done iteratively:

1. Form current approximate for $q()$.

2. Do one convergence step from the expected log. likelihood using that $q()$.

Mean Field, cont.

Approximate posterior on $\vec{l}$ and $\mathbf{v}$ by jointly independent $q(\vec{l}, \mathbf{v})$.

$$
\begin{aligned}
q(\vec{l}) &\longleftrightarrow \frac{1}{Z_l} \exp\left(E_{q(\mathbf{v})}\left\{\log p\left(\vec{l}, \mathbf{v}, \vec{w} \,|\, \mathbf{\Theta}, \vec{\alpha}, \vec{\beta}, K\right)\right\}\right) \\
q(\mathbf{v}) &\longleftrightarrow \frac{1}{Z_v} \exp\left(E_{q(\vec{l})}\left\{\log p\left(\vec{l}, \mathbf{v}, \vec{w} \,|\, \mathbf{\Theta}, \vec{\alpha}, \vec{\beta}, K\right)\right\}\right) \ ,
\end{aligned}
$$

Functional form is therefore:

$$
\begin{aligned}
l_k &\sim \text{Gamma}(a_k, b_k) \\
\{v_{j,k} : k = 1, \ldots, K\} &\sim \text{multinomial}(\{n_{j,k} : k = 1, \ldots, K\}, w_j)
\end{aligned}
$$

Mean Field, cont.

$$
\begin{aligned}
\vec{m} &\sim \text{Dirichlet}(\vec{a}) \\
\{v_{j,k} : k = 1, \ldots, K\} &\sim \text{multinomial}(\{n_{j,k} : k = 1, \ldots, K\}, w_j)
\end{aligned}
$$

$$
\begin{aligned}
n_{j,k} &= \frac{1}{Z_j} \theta_{j,k} \exp\left(\widehat{\log m_k}\right) \ , \\
a_k &= \alpha_k + \sum_j w_j n_{j,k} \ , \\
\textbf{where } \widehat{\log m_k} &\equiv \Psi_0(a_k) - \Psi_0\left(\sum_k a_k\right) \ , \\
Z_j &\equiv \sum_k \theta_{j,k} \exp\left(\widehat{\log m_k}\right) \ .
\end{aligned}
$$

Discrete PCA Algorithm Issues, cont.

- Can estimate $\vec{\alpha}$ from the data. Its posterior has a conjugate distribution to a $\Gamma(\alpha, 1)$, and is well-behaved. Roughly corresponds to setting α_k to the geometric mean* of the l_k's in the data.

- Can estimate $\vec{\beta}$ from the data. This has a gamma distribution, and is well-behaved. Corresponds to setting $\beta_k = \alpha_k / \langle l_k \rangle$.

- Can use Gibbs based "optimal importance sampler" to estimate document posterior

$$p(\vec{w} \,|\, \text{discrete PCA}, K, \vec{\alpha}, \vec{\beta}, \mathbf{\Theta})$$

"what if" questions about probability for additional words $\vec{w}_+$ in a document

$$p(\vec{w}_+ \,|\, \vec{w}, \text{discrete PCA}, K, \vec{\alpha}, \vec{\beta}, \mathbf{\Theta})$$

and true "evidence" for model comparisons

$$p(\vec{w}_1, \ldots, \vec{w}_I \,|\, \text{discrete PCA}, \vec{\alpha}, \vec{\beta}, K)$$

*mean taken in log space

Likelihood with Components Marginalised

Full likelihood for a document with hidden variables $\mathbf{v}$ is

$$p(\mathbf{v}, \vec{w} \,|\, \text{discrete PCA}, K, \vec{\alpha}, \vec{\beta}, \mathbf{\Theta}) \ .$$

This has components $\vec{l}$ marginalized out. In the gamma-Poisson case this becomes

$$\prod_k \frac{\beta_k^{\alpha_k}}{\Gamma(\alpha_k)} \prod_k \frac{\Gamma\left(\alpha_k + \sum_j v_{j,k}\right)}{(1+\beta_k)^{\alpha_k + \sum_j v_{j,k}}} \prod_j \prod_k \frac{\theta_{j,k}^{v_{j,k}}}{v_{j,k}!}$$

In the Dirichlet-discrete case this becomes

$$\frac{\Gamma\left(\sum_k \alpha_k + L_1\right)}{(\delta_P + \beta)^{\left(\sum_k \alpha_k + L_1\right)}} \frac{\prod_k \Gamma\left(\alpha_k + \sum_j v_{j,k}\right)}{\Gamma\left(\sum_k \alpha_k + L\right)} \prod_{g \in 2,\ldots,G} L_g! \prod_k \frac{\beta_k^{\alpha_k}}{\Gamma(\alpha_k)} \prod_j \prod_k \frac{\theta_{j,k}^{v_{j,k}}}{v_{j,k}!}$$

Gibbs with Rao-Blackwellisation

- In the above formula, can sample each words component assignment, thus for a word with index j, decrement v_{j,k_1} and increment v_{j,k_2}, and the probability for the change easily computed.

- Marginalising out variables in a problem prior to applying Gibbs sampling is called Rao-Blackwellisation.

- Rao-Blackwellisation sometimes significantly speeds things up.

- Previously formula showed how to marginalise out components $\vec{l}$ from a document probability. We can also marginalise out $\mathbf{\Theta}$ from the full joint likelihood to get a closed formula for

$$p(\mathbf{v}_1, \ldots, \mathbf{v}_I, \vec{w}_1, \ldots, \vec{w}_I \,|\, \text{discrete PCA}, \vec{\alpha}, \vec{\beta}, K)$$

- Rao-Blackwellisation applies to Gibbs for the gamma-Poisson case and the Dirichlet-discrete case*. This sampling involves $\mathbf{v}_1, \ldots, \mathbf{v}_I$ only.

*Griffiths and Steyvers, 2004

Estimating a "good" number of components K

- Like to estimate $p(\vec{w}_1, \ldots, \vec{w}_I \,|\, \text{discrete PCA}, \vec{\alpha}, \vec{\beta}, K)$, but we are sampling over $\mathbf{\Theta}$, $\mathbf{v}_1, \ldots, \mathbf{v}_I$, $\vec{w}_1, \ldots, \vec{w}_I$, etc.
- To estimate $p(\vec{x}) = \int_\phi p(\vec{x}|\phi)p(\phi)\mathrm{d}\,\phi$, when we are sampling sequence $\phi_{(1)}, \phi_{(2)}, \ldots, \phi_{(N)}$ according to posterior $p(\vec{x}|\phi)p(\phi)$, use

$$p(\vec{x}) \;\approx\; \frac{N}{\sum_n 1/p(\vec{x}|\phi_{(n)})} \;.$$

- This can be shown to be optimal importance sampling (if Gibbs were not used to generate samples).
- Apply above formula with hidden variables and parameters on the RHS: $p(\vec{w}_1, \ldots, \vec{w}_I \,|\, \text{discrete PCA}, \vec{\alpha}, \vec{\beta}, \mathbf{\Theta}, \mathbf{v}_1, \ldots, \mathbf{v}_I, \vec{w}_1, \ldots, \vec{w}_I, K)$
- Examples
  - $I = 1{,}800$ VLDB abstracts with $J = 5{,}000$ word vocabulary yields $K = 16$ as optimum.
  - $I = 20{,}000$ 20-newsgroups corpus with $J = 30{,}000$ word vocabulary yields $K = 200$ as optimum (± 20).

Comparative Performance: Algorithms

- Use a Dirichlet-discrete model with $\beta = 1$ and $\alpha_k = 0.5$.
- A Fisher Dirichlet prior is used for documents.
- Perplexity results are measured on a hold-out set.
- Algorithms:

  Mean field with inner loop convergence: As done in Minka's mean field implementation available in Matlab, reported by some. Inner loop of the mean field algorithm repeated to convergence on first cycle.

  Mean field: As described.

  Direct Gibbs: As described.

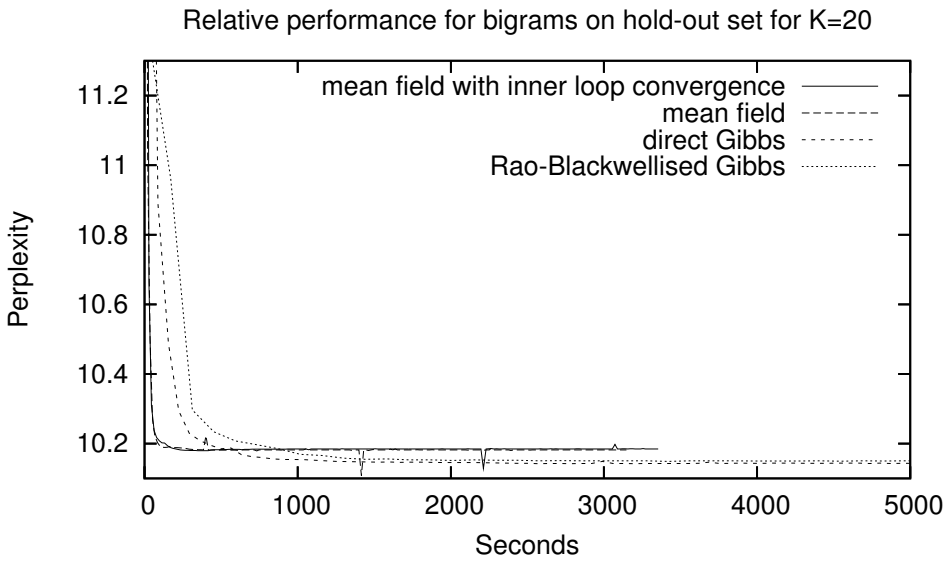
  Rao-Blackwellised Gibbs: With adjustments to ensure unbiases estimates of the log-probabilities. This adds about 50% time penalty, thus time reported below discounts this.

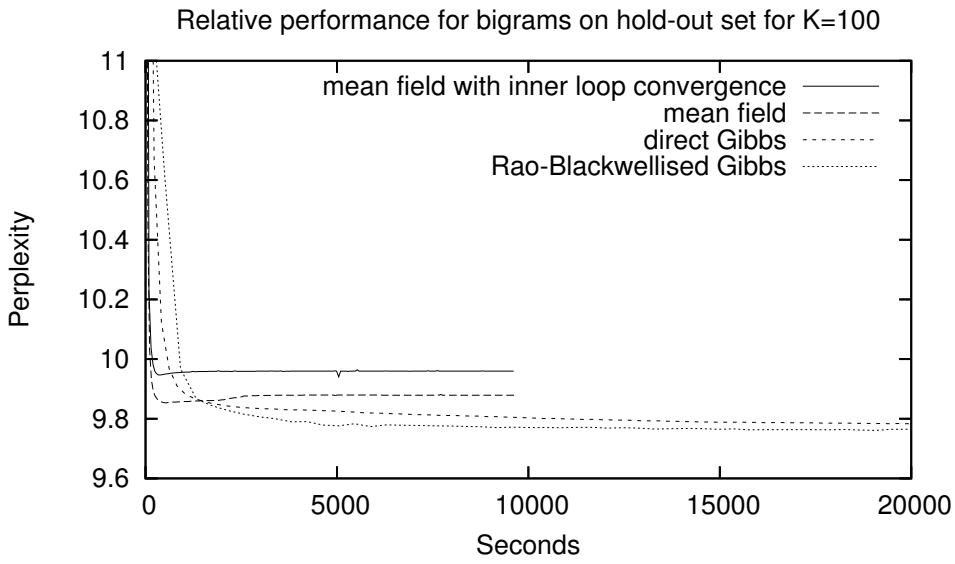
Comparative Performance: Data

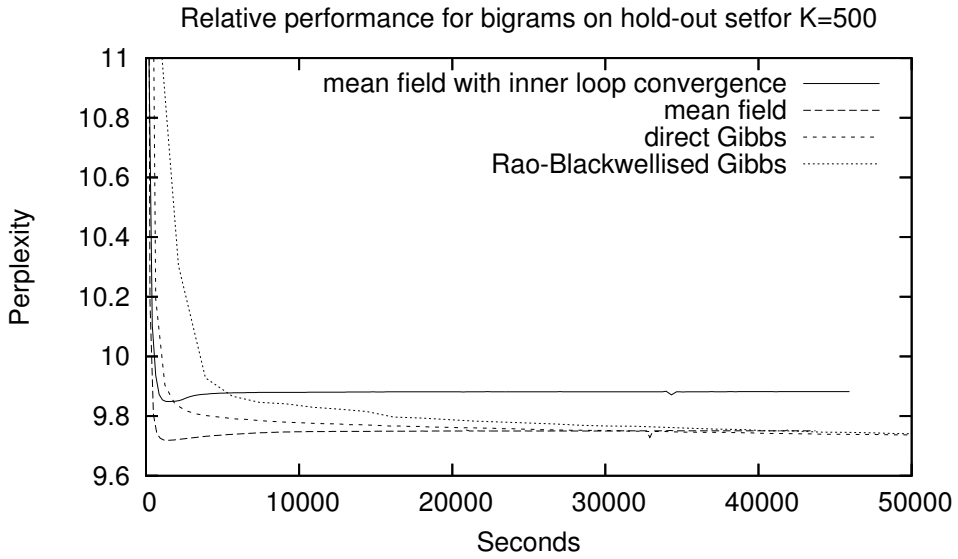
Data from the Reuters RCV1 corpus (Volume 1: English Language, 1996-08-20 to 1997-08-19).

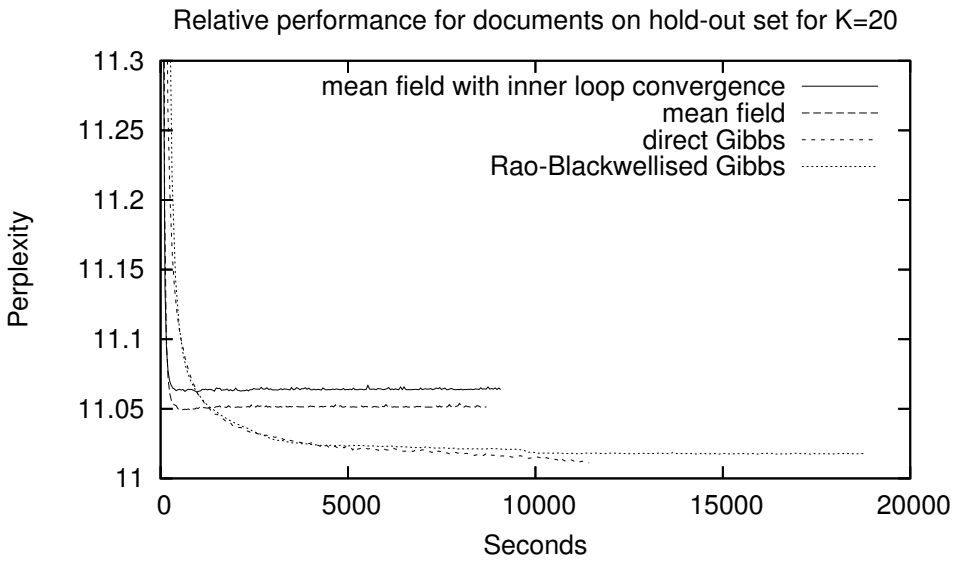
Bigrams: verb-noun bigrams in sentences, extracted from a lemmatized version of the full corpus. Single "document" has all noun lemma used in the vicinity of a given verb lemma. training $I = 30000$, $J = 20000$, and $K = 20$, 100, 500, $S/I \approx 700$. Testing is on random selection of $20,000$ held out.

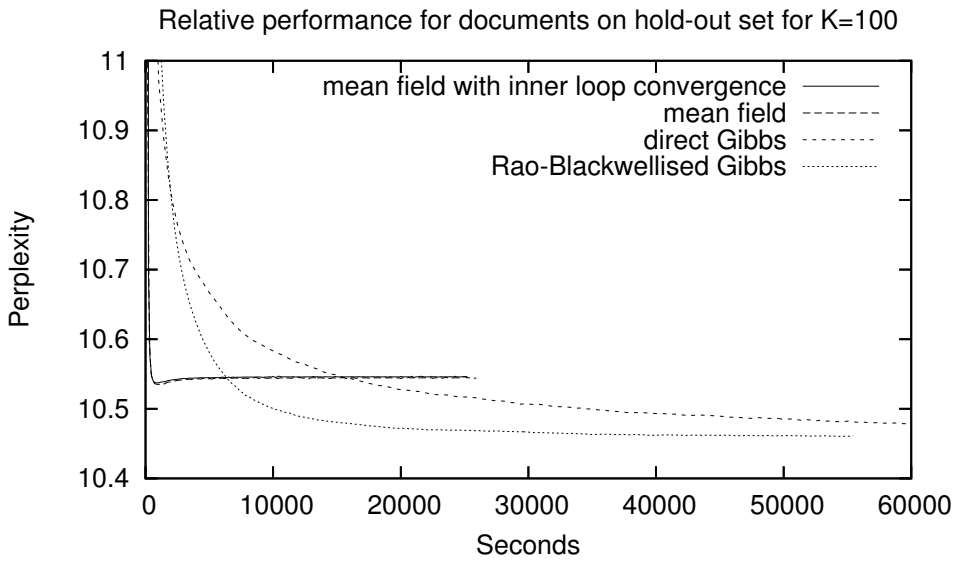
Documents: uses the first $150,000$ documents in the corpus, but only records the most frequent $50,000$ lemmatised nouns extracted from a lemmatized version of the corpus. training $I = 100000$, $J = 50000$, and $K = 20$, 100, $S/I \approx 100$. Testing on last 50000.


Relative performance for bigrams on hold-out set for K=20
mean field with inner loop convergence
mean field
direct Gibbs
Rao-Blackwellised Gibbs
Perplexity
Seconds
11.2
11
10.8
10.6
10.4
10.2
0
1000
2000
3000
4000
5000


Relative performance for bigrams on hold-out set for K=100
mean field with inner loop convergence
mean field
direct Gibbs
Rao-Blackwellised Gibbs
Perplexity
Seconds
11
10.8
10.6
10.4
10.2
10
9.8
9.6
0
5000
10000
15000
20000


Relative performance for bigrams on hold-out setfor K=500
Perplexity
Seconds
mean field with inner loop convergence
mean field
direct Gibbs
Rao-Blackwellised Gibbs
11
10.8
10.6
10.4
10.2
10
9.8
9.6
0
10000
20000
30000
40000
50000


Relative performance for documents on hold-out set for K=20
Perplexity
Seconds
mean field with inner loop convergence
mean field
direct Gibbs
Rao-Blackwellised Gibbs
11.3
11.25
11.2
11.15
11.1
11.05
11
0
5000
10000
15000
20000


Relative performance for documents on hold-out set for K=100
Perplexity
Seconds
mean field with inner loop convergence
mean field
direct Gibbs
Rao-Blackwellised Gibbs
11
10.9
10.8
10.7
10.6
10.5
10.4
0
10000
20000
30000
40000
50000
60000

Results summary

- Rao-Blackwellised Gibbs version has the fastest convergence (unless unbiased estimates of log-probabilities are required).
- Doing inner convergence of the loop in the mean field algorithm can damage performance.
- Mean field converges faster, and sometimes much faster. In one case, mean field also equals the others in perplexity.
- The direct Gibbs version usually catches up with the Rao-Blackwellised Gibbs version over time.
- The mean field version appears to start over-fitting quite quickly.