Discrete Principal Component Analysis*

Wray Buntine[1] and Aleks Jakulin[2]

[1] Complex Systems Computation Group (CoSCo),
Helsinki Institute for Information Technology (HIIT)
P.O. Box 9800, FIN-02015 HUT, Finland.
Wray.Buntine@hiit.fi
[2] Department of Knowledge Technologies
Jozef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
jakulin@gmail.com

Abstract. This article presents a unified theory for analysis of components in discrete data, and compares the methods with techniques such as independent component analysis (ICA), non-negative matrix factorisation (NMF) and latent Dirichlet allocation (LDA). The main families of algorithms discussed are mean field, Gibbs sampling, and Rao-Blackwellised Gibbs sampling. Applications are presented for voting records from the United States Senate for 2003, and the use of components in subsequent classification.

1 Introduction

Principal component analysis (PCA), latent semantic indexing (LSI), and independent component analysis (ICA, see [19]) are key methods in the statistical engineering toolbox. They have a long history, are used in many different ways, and under different names. They were primarily developed in the engineering community where the notion of a filter is common, and maximum likelihood methods less so. They are usually applied to measurements and real valued data, and used for feature extraction or data summarization.

Relatively recently the statistical computing community has become aware of seemingly similar approaches for discrete data that appears under many names: grade of membership (GOM) [31], probabilistic latent semantic indexing (PLSI) [17], non-negative matrix factorisation (NMF) [23], genotype inference using admixtures [29], latent Dirichlet allocation (LDA) [5], multinomial PCA (MPCA) [6], multiple aspect modelling [26], and Gamma-Poisson models (GaP) [9]. We refer to these methods jointly as *Discrete PCA* (DPCA), and this article provides a unifying model for them. Note also, that it is possible these methods existed in reduced form in other statistical fields perhaps decades earlier.

These methods are applied in the social sciences, demographics and medical informatics, genotype inference, text and image analysis, and information

* Draft paper for the PASCAL Bohinj workshop. Cite as HIIT Technical Report, July 2005.

retrieval. By far the largest body of applied work in this area (using citation indexes) is in genotype inference due to the Structure program [29]. A growing body of work is in text classification and topic modelling (see [16, 8]), and language modelling in information retrieval (see [1, 7, 9]).

Here we present in Section 3 a unified theory for analysis of components in discrete data, and compare the methods with related techniques. The main families of algorithms discussed are mean field, Gibbs sampling, and Rao-Blackwellised Gibbs sampling in Section 5. Applications are presented in Section 6 for voting records from the United States Senate for 2003, and the use of components in subsequent classification.

2 Views of DPCA

One interpretation of the DPCA methods is that they are a way of approximating large sparse discrete matrices. Suppose we have a $500,000$ documents made up of $1,500,000$ different words. A document such as a page out of Dr. Zeuss's *The Cat in The Hat*, is first given as a *sequence of words*.

> So, as fast as I could, I went after my net. And I said, "With my net I can bet them I bet, I bet, with my net, I can get those Things yet!"

It can be put in the *bag of words* representation, where word order is lost, as a list of words and their counts in brackets:

> so (1) as (2) fast (1) I (7) could (1) went (1) after (1) my (3) net (3) and (1) said (1) with (2) can (2) bet (3) them (1) get (1) those (1) things (1) yet (1) .

This sparse vector can be represented as a vector in full word space with $1,499,981$ zeroes and the counts above making the non-zero entries in the appropriate places. Given a matrix made up of rows of such vectors of non-negative integers dominated by zeros, it is called here a *large sparse discrete matrix*.

Bag of words is a basic representation in information retrieval [2]. The alternative is sequence of words. In DPCA, either representation can be used and the models act the same, up to any word order effects introduced by incremental algorithms. This detail is made precise in subsequent sections.

In this section, we argue from various perspectives that large sparse discrete data is not well suited to standard PCA, which is implicitly Gaussian, or ICA methods, which are justified using continuous data.

2.1 DPCA as Approximation

In the PCA method, one seeks to approximate a large matrix by a product of smaller matrices, by eliminating the lower-order eigenvectors, the least contributing components in the least squares sense. This is represented in Figure 1. If there are I documents, J words and K components, then the matrix on the

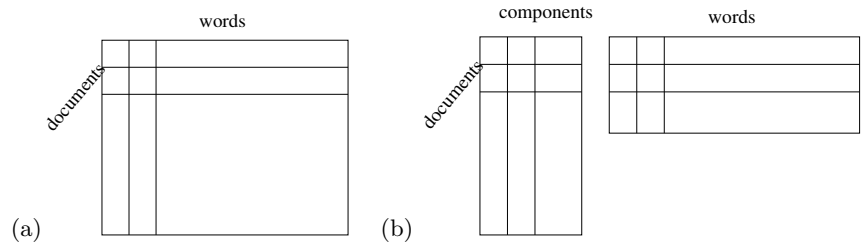

Fig. 1. The Matrix View: matrix product (b) approximates sparse discrete matrix (a)

left has $I * J$ entries and the two matrices on the right have $(I + J) * K$ entries. This represents a simplification when $K \ll I, J$. We can view DPCA methods as seeking the same goal in the case where the matrices are sparse and discrete.

When applying PCA to large sparse discrete matrices, interpretation becomes difficult. Negative values appear in the component matrices, so they cannot be interpreted as "typical documents" in any usual sense. This applies to many other kinds of sparse discrete data: low intensity images (such as astronomical images) and verb-noun data used in language models introduced by [27], for instance. DPCA, then, places constraints on the approximating matrices in Figure 1(b) so that they are also non-negative.

Also, there are fundamentally two different kinds of large sample approximating distributions that dominate discrete statistics: the Poisson and the Gaussian. For instance, a large sample binomial is approximated as a Poisson when the probability is small and as a Gaussian otherwise [30]. Thus in image analysis based on analogue to digital converters, where data is counts, Gaussian errors can sometimes be assumed, but the Poisson should be used if counts are small. DPCA then avoids Gaussian modelling of the data.

2.2 Independent Components

Independent component analysis (ICA) was also developed as an alternative to PCA. Hyvänen and Oja [20] argue that PCA methods merely decorrelate data, finding uncorrelated components. ICA then was developed as a way of representing multivariate data with truly independent components. The basic formulation is that a K-dimensional data vector $\boldsymbol{w}$ is a linear function of K independent components represented as a K-dimensional hidden vector $\boldsymbol{l}$, $\boldsymbol{w} = \boldsymbol{\Theta l}$ for square matrix $\boldsymbol{\Theta}$. For some univariate density function $g()$, the independent components are distributed as $p(\boldsymbol{l}) = \prod_k g(l_k)$. This implies

$$p(\boldsymbol{w} \,|\, \boldsymbol{\Theta}) \;=\; \prod_k g\left((\boldsymbol{\Theta w})_k\right) \frac{1}{\det(\boldsymbol{\Theta})}$$

The Fast ICA algorithm [20] can be interpreted as a maximum likelihood approach based on this probability formula. In the sparse discrete case, however,

this formulation breaks down for the simple reason that $\boldsymbol{w}$ is mostly zeros: the equation can only hold if $\boldsymbol{l}$ and $\boldsymbol{\Theta}$ are discrete as well and thus the gradient-based algorithms for ICA cannot be justified. To get around this in practice, when applying ICA to documents [4], word counts are sometimes first turned into TF-IDF scores [2].

To arrive at a formulation more suited to discrete data, we can relax the equality in ICA (i.e., $\boldsymbol{w} = \boldsymbol{\Theta l}$) to be an expectation:

$$\mathrm{Exp}_{p(\boldsymbol{w}|\boldsymbol{l})}(\boldsymbol{w}) \;=\; \boldsymbol{\Theta l}\ .$$

We still have independent components, but a more general relationship then exists with the data. Correspondence between ICA and DPCA has been reported in [7, 9]. Note with this expectation relationship, the dimension of $\boldsymbol{l}$ can now be less than the dimension of $\boldsymbol{w}$, and thus $\boldsymbol{\Theta}$ would be a rectangular matrix.

For $p(\boldsymbol{w} \mid \boldsymbol{\Theta})$ in the so-called exponential family distributions [15], the expected value of $\boldsymbol{w}$ is referred to as the *dual parameter*, and it is usually the parameter we know best. For the Bernoulli with probability p, the dual parameter is p, for the Poisson with rate λ, the dual parameter is λ, and for the Gaussian with mean μ, the dual parameter is the mean. Our formulation, then, can be also be interpreted as letting $\boldsymbol{w}$ be exponential family with dual parameter given by $(\boldsymbol{\Theta l})$. Our formulation then generalises PCA in the same way that linear models [25] generalises linear regression[3].

Note, an alternative has also been presented [13] where $\boldsymbol{w}$ has an exponential family distribution with natural parameters given by $(\boldsymbol{\Theta l})$. For the Bernoulli with probability p, the natural parameter is $\log p/(1-p)$, for the Poisson with rate λ, the natural parameter is $\log \lambda$ and for the Gaussian with mean μ, the natural parameter is the mean. This formulation generalises PCA in the same way that generalised linear models [25] generalises linear regression.

3 The Basic Model

A good introduction to these models from a number of viewpoints is by [5, 9, 7]. Here we present a general model and show its relationship to previous work. The notation of words, bags and documents will be used throughout, even though other kinds of data representations also apply. In machine learning terminology, a word is a feature, a bag is a data vector, and a document is a sample. Notice that the bag collects the words in the document and loses their ordering. The bag is represented as a data vector $\boldsymbol{w}$. It is now J-dimensional, and the hidden vector $\boldsymbol{l}$ called the *components* is K-dimensional, where $K < J$. The term *component* is used here instead of, for instance, topic. The parameter matrix $\boldsymbol{\Theta}$ is $J \times K$.

3.1 Bags or Sequences of Words?

For a document d represented as a sequence of words, if $\boldsymbol{w} = \mathrm{bag}(d)$ is its bagged form, the bag of words, represented as a vector of counts, then for any model $\mathcal{M}$

[3] Though it is an artifact of statistical methods in general that linear models are best defined as generalised linear models with a linear link function.

that consistently handles both kinds of data

$$p(\boldsymbol{w} \,|\, \mathcal{M}) \;=\; \frac{(\sum w_j)!}{\prod_j w_j!} p(d \,|\, \mathcal{M}) \;. \tag{1}$$

Note that some likelihood based methods such as maximum likelihood, some Bayesian methods, and some other fitting methods (for instance, a cross validation technique) use the likelihood as a functional. They take values or derivatives but otherwise do not further interact with the likelihood. The combinatoric first term on the right hand side of Equation (1) can be dropped here safely because it does not affect the fitting of the parameters for $\mathcal{M}$. Thus for these methods, it is irrelevant whether the data is treated as a bag or as a sequence. This is a general property of multinomial data.

Some advanced fitting methods such as Gibbs sampling do not treat the likelihood as a functional. They introduce hidden variables that changes the likelihood, and they may update parts of a document in turn. For these, ordering affects can be incurred by bagging a document, since updates for different parts of the data will now be done in a different order. But the combinatoric first data dependent term in Equation (1) will still be ignored and the algorithms are effectively the same up to the ordering affects. *Thus, while we consider bag of words in this article, all the theory equally applies to the sequence of words representation. Implementation can easily address both cases with little change to the algorithms, just to the data handling routines.*

3.2 General DPCA

The general formulation introduced in Section 2.2 is an unsupervised version of a linear model, and it applies to the bag of words $\boldsymbol{w}$ (the sequence of words version replaces $\boldsymbol{w}$ with d) as

$$\begin{aligned} \boldsymbol{w} &\sim \mathrm{PD}_D(\boldsymbol{\Theta l}, \boldsymbol{\alpha}_1) \;\text{ such that }\; \mathrm{Exp}_{p(\boldsymbol{w}|\boldsymbol{l})}(\boldsymbol{w}) \;=\; \boldsymbol{\Theta l} \\ \boldsymbol{l} &\sim \mathrm{PD}_C(\boldsymbol{\alpha}_2) \;, \end{aligned} \tag{2}$$

where $\mathrm{PD}_D(\cdot,\cdot)$ is a discrete vector valued probability distribution, $\mathrm{PD}_C(\cdot)$ is a continuous vector valued probability density function that may introduce constraints on the domain of $\boldsymbol{l}$ such as non-negativity. The hyper-parameter vectors $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ might also be in use. In the simplest case these two distributions are a product of univariate distributions, so we might have $l_k \sim \mathrm{PD}_c(\alpha_k)$ for each k as in Section 2.2.

Various cases of DPCA can now be represented in terms of this format, as given in Table 1. The *discrete* distribution is the multivariate form of a binomial where an index $j \in \{0, 1, ..., J-1\}$ is selected according to a J-dimensional probability vector. A multinomial with total count L is the bagged version of L discretes. In the table, *non-spec* indicates that this aspect of the model was left unspecified. Note that NMF used a cost function formulation, and thus avoided defining likelihood models. PLSI used a weighted likelihood formulation that treated hidden variables without defining a distribution for them.

Table 1. DPCA Example Models

Name	Bagged	PD_D	PD_C	Constraints
NMF [23]	yes	*non-spec*	*non-spec*	$l_k \geq 0$
PLSI [17]	no	discrete	*non-spec*	$l_k \geq 0$, $\sum_k l_k = 1$
LDA [5]	no	discrete	Dirichlet	$l_k \geq 0$, $\sum_k l_k = 1$
MPCA [6]	yes	multinomial	Dirichlet	$l_k \geq 0$, $\sum_k l_k = 1$
GaP [9]	yes	Poisson	gamma	$l_k \geq 0$

The formulation of Equation (2) is also called an *admixture* in the statistical literature [29]. Let $\boldsymbol{\theta}_k$ be the k-th row of $\boldsymbol{\Theta}$. The parameters for the distribution of $\boldsymbol{w}$ are formed by making a weighted sum of the parameters represented by the K vectors $\boldsymbol{\theta}_k$. This is in contrast with a *mixture model* [15] which is formed by making a weighted sum of the probability distributions $\mathrm{PD}_D(\boldsymbol{\theta}_k, \boldsymbol{\alpha}_1)$.

3.3 The Gamma-Poisson Model

The general Gamma-Poisson form of DPCA, introduced as GaP [9] is now considered in more detail:

- Document data is supplied in the form of *word counts*. The word count for each word type is w_j. Let L be the total count, so $L = \sum_j w_j$.
- The document also has *hidden component variables* l_k that indicate the amount of the component in the document. In the general case, the l_k are independent and gamma distributed

$$l_k \;\sim\; \mathrm{Gamma}(\alpha_k, \beta_k) \qquad\qquad \text{for } k = 1, \ldots, K.$$

  The β_k affects scaling of the components[4], while α_k changes the shape of the distribution.
- Initially the K-dimensional hyper-parameter vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ will be treated as constants, and their estimation will be considered later.
- There is a *general model matrix* $\boldsymbol{\Theta}$ of size $J \times K$ with entries $\theta_{j,k}$ that controls the partition of features amongst each component. This is normalised along rows ($\sum_j .\theta_{j,k} = 1$).
- The data is now Poisson distributed, for each j

$$w_j \;\sim\; \mathrm{Poisson}\left((\boldsymbol{\Theta}\boldsymbol{l})_j\right) \;.$$

The hidden or latent variables here are the vector $\boldsymbol{l}$. The model parameters are the gamma hyper-parameters $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$, and the matrix $\boldsymbol{\Theta}$. The full likelihood for each document then becomes:

$$\prod_k \frac{\beta_k^{\alpha_k} l_k^{\alpha_k - 1} e^{-\beta_k l_k}}{\Gamma(\alpha_k)} \prod_j \frac{\left(\sum_k l_k \theta_{j,k}\right)^{w_j} e^{-\left(\sum_k l_k \theta_{j,k}\right)}}{w_j!} \tag{3}$$

[4] Note conventions for the gamma vary. Sometimes a parameter $1/\beta_k$ is used. Our convention is revealed in Equation (3).

To obtain a full posterior for the problem, these terms are multiplied across all documents, and a Dirichlet prior for $\boldsymbol{\Theta}$ added in.

3.4 The Dirichlet-Multinomial Model

The Dirichlet-multinomial form of DPCA was introduced as MPCA. In this case, the normalised hidden variables $\boldsymbol{m}$ are used, and the total word count L is not modelled.

$$\boldsymbol{m} \sim \text{Dirichlet}_K(\boldsymbol{\alpha}) , \qquad \boldsymbol{w} \sim \text{Multinomial}(\boldsymbol{\Theta m}, L)$$

The full likelihood for each document then becomes:

$$C_{\boldsymbol{w}}^{L} \Gamma\left(\sum_k \alpha_k\right) \prod_k \frac{m_k^{\alpha_k - 1}}{\Gamma(\alpha_k)} \prod_j \left(\sum_k l_k \theta_{j,k}\right)^{w_j} \tag{4}$$

where $C_{\boldsymbol{w}}^{L}$ is L choose $\boldsymbol{w}$. LDA has the multinomial replaced by a discrete, and thus the choose term drops, as per Section 3.1. PLSI is related to LDA but lacks a prior distribution on $\boldsymbol{m}$. It does not model these hidden variables using probability theory, but instead using a weighted likelihood method [17].

4 Aspects of the Model

Before starting analysis and presenting algorithms, this section discusses some useful aspects of the DPCA model family.

4.1 Correspondence

While it was argued in Section 3.4 that LDA and MPCA are essentially the same model, up to ordering effects[5] it is also the case that these methods have a close relationship to GaP.

Consider the Gamma-Poisson model. A set of Poisson variables can always be converted into a total Poisson and a multinomial over the set [30]. The set of Poisson distributions on w_j then is equivalent to:

$$L \sim \text{Poisson}\left(\sum_k l_k\right) , \qquad \boldsymbol{w} \sim \text{Multinomial}\left(\frac{1}{\sum_k l_k} \boldsymbol{\Theta l}, L\right) .$$

Moreover, if the $\boldsymbol{\beta}$ hyper-parameter is a constant vector, then $m_k = l_k / \sum_k l_k$ is distributed as a Dirichlet, $\text{Dirichlet}_K(\boldsymbol{\alpha})$, and $\sum_k l_k$ is distributed independently as a gamma $\Gamma(\sum_k \alpha_k, \beta)$. The second distribution above can then be represented as

$$\boldsymbol{w} \sim \text{Multinomial}\left(\boldsymbol{\Theta m}, L\right) .$$

Note also, that marginalising out $\sum_k l_k$ convolves a Poisson and a gamma distribution to produce a Poisson-Gamma distribution for L [3]. Thus, we have proven the following.

[5] Note that the algorithm for MPCA borrowed the techniques of LDA directly.

Lemma 1. *Given a Gamma-Poisson model of Section 3.3 where the $\boldsymbol{\beta}$ hyper-parameter is a constant vector with all entries the same, β, the model is equivalent to a Dirichlet-multinomial model of Section 3.4 where $m_k = l_k / \sum_k l_k$, and in addition*

$$L \;\sim\; \textit{Poisson-Gamma}\left(\sum_k \alpha_k, \beta, 1\right)$$

The implication is that if $\boldsymbol{\alpha}$ is treated as known, and not estimated from the data, then L is *aposteriori* independent of $\boldsymbol{m}$ and $\boldsymbol{\Theta}$. In this context, when the $\boldsymbol{\beta}$ hyper-parameter is a constant vector, LDA, MPCA and GaP are equivalent models ignoring representational issues. If $\boldsymbol{\alpha}$ is estimated from the data in GaP, then the presence of the data L will influence $\boldsymbol{\alpha}$, and thus the other estimates such as of $\boldsymbol{\Theta}$. In this case, LDA and MPCA will no longer be effectively equivalent to GaP. Note, Canny recommends fixing $\boldsymbol{\alpha}$ and estimating $\boldsymbol{\beta}$ from the data [9].

To complete the set of correspondences, note that in Section 5.1 it is proven that NMF corresponds to a maximum likelihood version of GaP, and thus it also corresponds to a maximum likelihood version of LDA, MPCA, and PLSI. To relationship is made by normalising the matrices that result.

4.2 A Multivariate Version

Another variation of the methods is to allow grouping of the count data. Words can be grouped into separate variable sets. These groups might be "title words," "body words," and "topics" in web page analysis or "nouns," "verbs" and "adjectives" in text analysis. The groups can be treated with separate discrete distributions, as below. The J possible word types in a document are partitioned into G groups $B_1, \ldots, B_G$. The total word counts for each group g is denoted $L_g = \sum_{j \in B_g} w_j$. If the vector $\boldsymbol{w}$ is split up into G vectors $\boldsymbol{w}_g = \{w_j : j \in B_g\}$, and the matrix $\boldsymbol{\Theta}$ is now normalised by group in each row, so $\sum_{j \in B_g} \theta_{j,k} = 1$, then a multivariate version of MPCA is created so that for each group g,

$$\boldsymbol{w}_g \;\sim\; \text{Multinomial}\left(\left\{\sum_k m_k \theta_{j,k} \;:\; j \in B_g\right\}, L_g\right) \;.$$

Fitting and modelling methods for this variation are related to LDA or MPCA, and will not be considered in more detail here. This has the advantage that different kinds of words have their own multinomial and the distribution of different kinds is ignored. This version is demonstrated subsequently on US Senate voting records, where each multinomial is now a single vote.

4.3 Component Assignments for Words

In standard mixture models, each document in a collection is assigned to one (hidden) component. In the DPCA family of models, each word in each document is assigned to one (hidden) component. To see this, we introduce another hidden vector which represents the component assignments for different words.

As in Section 3.1, this can be done using a bag of components or a sequence of components representation, and no effective change occurs in the basic models, or in the algorithms so derived. What this does is expand out the term $\boldsymbol{\Theta l}$ into parts, treating it as if it is the result of marginalising out some hidden variable. Here we just treat the two cases of Gamma-Poisson and Dirichlet-multinomial.

We introduce a K-dimensional discrete hidden vector $\boldsymbol{c}$ whose total count is L, the same as the word count. This records the component for each word in a bag of components representation. Thus c_k gives the number of words in the document appearing in the k-th component. Its posterior mean makes an excellent diagnostic and interpretable result. A document from the sports news might have 50 "football" words, 10 "German" words, 20 "sports fan" words and 20 "general vocabulary" words. So an estimate or expected value for $v_{\cdot,k}$ is thus used to give the amount of a component in a document, and forms the basis of any posterior analysis and hand inspection of the results.

With the new hidden vector, the distributions underlying the Gamma-Poisson model become:

$$\begin{aligned} l_k &\sim \text{Gamma}(\alpha_k, \beta_k) \\ c_k &\sim \text{Poisson}\,(l_k) \\ w_j &= \sum_k v_{j,k} \qquad \text{where } v_{j,k} \sim \text{Multinomial}\,(\boldsymbol{\theta}_k, c_k)\ . \end{aligned} \tag{5}$$

The joint likelihood thus becomes, after some rearrangement

$$\prod_k \frac{\beta_k^{\alpha_k} l_k^{c_k+\alpha_k-1} e^{-(\beta_k+1)l_k}}{\Gamma(\alpha_k)} \prod_{j,k} \frac{\theta_{j,k}^{v_{j,k}}}{v_{j,k}!}\ . \tag{6}$$

Note that $\boldsymbol{l}$ can be marginalised out, yielding

$$\prod_k \frac{\Gamma(c_k+\alpha_k)}{\Gamma(\alpha_k)} \frac{\beta_k^{\alpha_k}}{(\beta_k+1)^{c_k+\alpha_k}} \prod_{j,k} \frac{\theta_{j,k}^{v_{j,k}}}{v_{j,k}!} \tag{7}$$

and the posterior mean of l_k given $\boldsymbol{c}$ is $(c_k+\alpha_k)/(1+\beta_k)$. Thus each $c_k \sim$ Poisson-Gamma$(\alpha_k, \beta_k, 1)$. For the Dirichlet-multinomial model, a similar reconstruction applies:

$$\begin{aligned} \boldsymbol{m} &\sim \text{Dirichlet}_K(\boldsymbol{\alpha}) \\ c_k &\sim \text{Multinomial}\,(\boldsymbol{m}, L) \\ w_j &= \sum_k v_{j,k} \qquad \text{where } v_{j,k} \sim \text{Multinomial}\,(\boldsymbol{\theta}_k, c_k)\ . \end{aligned} \tag{8}$$

The joint likelihood thus becomes, after some rearrangement

$$L!\,\Gamma\left(\sum_k \alpha_k\right) \prod_k \frac{m_k^{c_k+\alpha_k-1}}{\Gamma(\alpha_k)} \prod_{j,k} \frac{\theta_{j,k}^{v_{j,k}}}{v_{j,k}!}\ . \tag{9}$$

Again, $\boldsymbol{m}$ can be marginalised out yielding

$$L! \frac{\Gamma\left(\sum_k \alpha_k\right)}{\Gamma\left(L + \sum_k \alpha_k\right)} \prod_k \frac{\Gamma(c_k + \alpha_k)}{\Gamma(\alpha_k)} \prod_{j,k} \frac{\theta_{j,k}^{v_{j,k}}}{v_{j,k}!} \ . \tag{10}$$

Note, in both the Gamma-Poisson and Dirichlet-multinomial models each count w_j is the sum of counts in the individual components, $w_j = \sum_k v_{j,k}$. The sparse matrix $v_{j,k}$ of size $J \times K$ is thus a hidden variable, with row totals w_j given in the data and column totals c_k.

4.4 Historical Notes

The first clear enunciation of the large-scale model in its Poisson form comes from [23], and in its multinomial form from [17] and [29]. The first clear expression of the problem as a hidden variable problem is given by [29], although no doubt earlier versions appear in statistical journals under other names, and perhaps in reduced dimensions (e.g., $K = 3$). The application of Gibbs is due to [29], and of the mean field method is due to [5], the newer fast Gibbs version to [16]. The mean field method was a significant introduction because of its speed. The relationship between LDA and PLSI and that NMF was a Poisson version of LDA was first pointed out by [6]. The connections to ICA come from [7] and [9]. The general Gamma-Poisson formulation, perhaps the final generalisation to this line of work, is in [9].

5 Algorithms

Neither of the likelihoods yield to standard EM analysis. For instance, in the likelihood Equation (6) and (9), the hidden variables $\boldsymbol{l}/\boldsymbol{m}$ and $\boldsymbol{v}/\boldsymbol{c}$ are coupled, which is not compatible with the standard EM theory. Most authors point out that EM-like analysis applies and use EM terminology, and indeed this is the basis of the mean field approach for this problem. For the exponential family, mean field algorithms correspond to an extension of the EM algorithm [6]. EM and maximum likelihood algorithms can be used if one is willing to include the hidden variables in the optimization criterion, as done for instance with the K-means algorithm.

Algorithms for this problem follow some general approaches in the statistical computing community:

- Straight maximum likelihood such as by [23], sometimes expressed as a Kullback-Leibler divergence,
- annealed maximum likelihood by [17], best viewed in terms of its clustering precursor such as by [18],
- Gibbs sampling on $\boldsymbol{v}$, $\boldsymbol{l}/\boldsymbol{m}$ and $\boldsymbol{\Theta}$ in turn using a full probability distribution by [29], or Gibbs sampling on $\boldsymbol{v}$ alone (or equivalently, component assignments for words in the sequence of words representation) after marginalising out $\boldsymbol{l}/\boldsymbol{m}$ and $\boldsymbol{\Theta}$ by [16],

– mean field methods by [5], and
– expectation propagation (like so-called cavity methods) by [26].

The maximum likelihood algorithms are not considered here. Their use is always discouraged when a large number of hidden variables exist, and they are mostly simplifications of the algorithms here,. Expectation propagation does not scale, thus it is not considered subsequently either. While other algorithms typically require $O(L)$ or $O(K)$ hidden variables stored per document, expectation propagation requires $O(KL)$.

To complete the specification of the problem, a prior is needed for $\boldsymbol{\Theta}$. A Dirichlet prior can be used for each k-th component of $\boldsymbol{\Theta}$ with J prior parameters γ_j, so $\boldsymbol{\theta}_k \sim \text{Dirichlet}_J(\boldsymbol{\gamma})$.

5.1 Mean Field

The mean field method was first applied to the sequential variant of the Dirichlet-multinomial version of the problem by [5]. In the mean field approach, generally a factored posterior approximation $q()$ is developed for the hidden variables $\boldsymbol{l}$ (or $\boldsymbol{m}$) and $\mathbf{v}$, and it works well if it takes the same functional form as the prior probabilities for the hidden variables. The functional form of the approximation can be derived by inspection of the recursive functional forms (see [6] Equation (4)):

$$\begin{aligned} q(\boldsymbol{l}) &\longleftrightarrow \frac{1}{Z_l} \exp\left(E_{q(\mathbf{v})}\left\{\log p\left(\boldsymbol{l}, \mathbf{v}, \boldsymbol{w} \mid \boldsymbol{\Theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}, K\right)\right\}\right) \\ q(\mathbf{v}) &\longleftrightarrow \frac{1}{Z_v} \exp\left(E_{q(\boldsymbol{l})}\left\{\log p\left(\boldsymbol{l}, \mathbf{v}, \boldsymbol{w} \mid \boldsymbol{\Theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}, K\right)\right\}\right) \ , \end{aligned} \tag{11}$$

where the Z_l and Z_r are normalising constants.

An important computation used during convergence in this approach is a lower bound on the individual document log probabilities, $\log p\left(\boldsymbol{w} \mid \boldsymbol{\Theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}, K\right)$. This comes naturally out of the mean field algorithm (see [6] Equation (6)). Using the approximation $q()$ defined by the above distributions, the bound is given by

$$\text{Exp}_{q()}\left(\log\, p\left(\boldsymbol{l}, \mathbf{v}, \boldsymbol{w} \mid \boldsymbol{\Theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}, K\right)\right) + I(q()) \ . \tag{12}$$

The mean field algorithm applies to the Gamma-Poisson version and the Dirichlet-multinomial version.

Gamma-Poisson model: So for the Gamma-Poisson case introduce approximation parameters a_k, b_k and $n_{j,k}$ (normalised as $\sum_k n_{j,k} = 1$), and then use posterior approximations matching the corresponding model in Equations (5):

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

for each document. Matching the likelihood these two form with Equation (6), according to the rules of Equations (11), yields the re-write rules for these new parameters:

$$
\begin{aligned}
n_{j,k} &= \frac{1}{Z_j}\theta_{j,k}\exp\left(\widehat{\log l_k}\right) \ , \\
a_k &= \alpha_k + \sum_j w_j n_{j,k} \ , \\
b_k &= 1 + \beta_k \ , \\
\textbf{where } \widehat{\log l_k} &\equiv \mathrm{Exp}_{l_k \mid a_k . b_k}(\log\ l_k) \ = \ \Psi_0(a_k) - \log b_k \ , \\
Z_j &\equiv \sum_k \theta_{j,k}\exp\left(\widehat{\log l_k}\right) \ .
\end{aligned}
\tag{13}
$$

Here, $\Psi_0()$ is the digamma function. These equations form the first step of each major cycle, and are performed on each document.

The second step is to re-estimate the original model parameters $\boldsymbol{\Theta}$ using the posterior approximation by maximising the expectation according to $q(\boldsymbol{l}, \mathbf{v})$ of the log of the full posterior probability $\mathrm{Exp}_{q()}\left(\log\, p\left(\boldsymbol{l}, \mathbf{v}, \boldsymbol{w}, \boldsymbol{\Theta}, \mid \boldsymbol{\alpha}, \boldsymbol{\beta}, K\right)\right)$. This incorporates Equation (6) for each document, and a Dirichlet prior term for $\boldsymbol{\Theta}$. Denote the intermediate variables $\mathbf{n}$ for the i-th document by adding a (i) subscript, as $n_{j,k,(i)}$, and likewise for $w_{j,(i)}$. All these log probability formulas yield linear terms in $\theta_{j,k}$, thus one gets

$$
\theta_{j,k} \ \propto \ \sum_i w_{j,(i)} n_{j,k,(i)} + \gamma_j \ . \tag{14}
$$

The lower bound on the log probability of Equation (12), after some simplification and use of the rewrites of Equation (13), becomes

$$
\log \frac{1}{\prod_j w_j!} + \sum_k \left((\alpha_k - a_k)\widehat{\log l_k} - \log \frac{\Gamma(\alpha_k) b_k^{a_k}}{\Gamma(a_k)\beta_k^{\alpha_k}} \right) + \sum_j w_j \log Z_j \ . \tag{15}
$$

The mean field algorithm for Gamma-Poisson version is given in Figure 2.

Dirichlet-multinomial model: For the Dirichlet-multinomial case, the mean field algorithm takes a related form. The approximate posterior is given by:

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

This yields the same style update equations as Equations (13) except that $\beta_k = 1$

$$
\begin{aligned}
n_{j,k} &= \frac{1}{Z_j}\theta_{j,k}\exp\left(\widehat{\log m_k}\right) \ , \\
a_k &= \alpha_k + \sum_j w_j n_{j,k} \ ,
\end{aligned}
\tag{16}
$$

1. Initialise $\boldsymbol{a}$ for each document. The uniform initialisation would be $a_k = \left(\sum_k \alpha_k + L\right)/K$. Note $\mathbf{n}$ is not stored.
2. Do for each document:
   (a) Using Equations (13), recompute $\mathbf{n}$ and update $\boldsymbol{a}$ in place.
   (b) Concurrently, compute the log-probability bound of Equation (15), and add to a running total.
   (c) Concurrently, maintain the sufficient statistics for $\boldsymbol{\Theta}$, the total $\sum_i w_{j,(i)} n_{j,k,(i)}$ for each j, k over documents.
   (d) Store $\boldsymbol{a}$ for the next cycle and discard $\mathbf{n}$.
3. Update $\boldsymbol{\Theta}$ using Equation (14), normalising appropriately.
4. Report the total log-probability bound, and repeat, starting at Step 2.

Fig. 2. Mean Field Algorithm for Gamma-Poisson

$$\textbf{where } \widehat{\log m_k} \equiv \mathrm{Exp}_{m_k \mid \boldsymbol{a}}(\log\, m_k) \;=\; \Psi_0(a_k) - \Psi_0\left(\sum_k a_k\right) \;,$$
$$Z_j \equiv \sum_k \theta_{j,k} \exp\left(\widehat{\log m_k}\right) \;.$$

Equation (14) is also the same. The lower bound on the individual document log probabilities, $\log p\left(\boldsymbol{w} \mid \boldsymbol{\Theta}, \boldsymbol{\alpha}, K\right)$ now takes the form

$$\log\left(C_{\boldsymbol{w}}^{L} \frac{\Gamma\left(\sum_k \alpha_k\right)}{\Gamma\left(\sum_k a_k\right)}\right) + \sum_k \left((\alpha_k - a_k)\widehat{\log m_k} - \log\frac{\Gamma(\alpha_k)}{\Gamma(a_k)}\right) + \sum_j w_j \log Z_j \;. \tag{17}$$

The mean field algorithm for Dirichlet-multinomial version is related to that in Figure 2. Equations (16) replace Equations (13), Equation (17) replaces Equation (15), and the initialisation for a_k should be 0.5, a Jeffreys prior.

Complexity: Because Step 2(a) only uses words appearing in a document, the full Step 2 is $O(SK)$ in time complexity where S is the number of words in the full collection. Step 3 is $O(JK)$ in time complexity. Space complexity is $O(IK)$ to store the intermediate parameters $\boldsymbol{a}$ for each document, and the $O(2JK)$ to store $\boldsymbol{\Theta}$ and its statistics. In implementation, Step 2 for each document is often quite slow, and thus both $\boldsymbol{a}$ and the document word data can be stored on disk and streamed, thus the main memory complexity is $O(2JK)$.

Correspondence with NMF A precursor to the GaP model is non-negative matrix factorisation (NMF) [24], which is based on the matrix approximation paradigm using Kullback-Leibler divergence. The algorithm itself, converted to the notation used here, is as follows

$$l_{k,(i)} \longleftarrow l_{k,(i)} \sum_j \frac{\theta_{j,k}}{\sum_j \theta_{j,k}} \frac{w_{j,(i)}}{\sum_k \theta_{j,k} l_{k,(i)}} \qquad \theta_{j,k} \longleftarrow \theta_{j,k} \sum_i \frac{l_{k,(i)}}{\sum_i l_{k,(i)}} \frac{w_{j,(i)}}{\sum_k \theta_{j,k} l_{k,(i)}}$$

Notice that the solution is indeterminate up to a factor ψ_k. Multiple $l_{k,(i)}$ by ψ_k and divide $\theta_{j,k}$ by ψ_k and the solution still holds. Thus, without loss of generality, let $\theta_{j,k}$ be normalised on j, so that $\sum_j \theta_{j,k} = 1$.

Lemma 2. *Take a solution to the NMF equations, and divide $\theta_{j,k}$ by a factor $\psi_k = \sum_j \theta_{j,k}$, and multiply $l_{k,(i)}$ by the same factor. This is equivalent to a solution for the following rewrite rules*

$$l_{k,(i)} \longleftarrow l_{k,(i)} \sum_j \theta_{j,k} \frac{w_{j,(i)}}{\sum_k \theta_{j,k} l_{k,(i)}} \qquad \theta_{j,k} \propto \theta_{j,k} \sum_i l_{k,(i)} \frac{w_{j,(i)}}{\sum_k \theta_{j,k} l_{k,(i)}}$$

where $\theta_{j,k}$ is kept normalised on j.

Proof. First prove the forward direction. Take the scaled solution to NMF. The NMF equation for $l_{k,(i)}$ is equivalent to the equation for $l_{k,(i)}$ in the lemma. Take the NMF equation for $\theta_{j,k}$ and separately normalise both sides. The $\sum_i l_{k,(i)}$ term drops out and one is left with the equation for $\theta_{j,k}$ in the lemma. Now prove the backward direction. It is sufficient to show that the NMF equations hold for the solution to the rewrite rules in the lemma, since $\theta_{j,k}$ is already normalised. The NMF equation for $l_{k,(i)}$ clearly holds. Assuming the rewrite rules in the lemma hold, then

$$\begin{aligned}
\theta_{j,k} &= \frac{\theta_{j,k} \sum_i l_{k,(i)} \frac{w_{j,(i)}}{\sum_k \theta_{j,k} l_{k,(i)}}}{\sum_j \theta_{j,k} \sum_i l_{k,(i)} \frac{w_{j,(i)}}{\sum_k \theta_{j,k} l_{k,(i)}}} \\
&= \frac{\theta_{j,k} \sum_i l_{k,(i)} \frac{w_{j,(i)}}{\sum_k \theta_{j,k} l_{k,(i)}}}{\sum_i l_{k,(i)} \sum_j \theta_{j,k} \frac{w_{j,(i)}}{\sum_k \theta_{j,k} l_{k,(i)}}} && \text{(reorder sum)} \\
&= \frac{\theta_{j,k} \sum_i l_{k,(i)} \frac{w_{j,(i)}}{\sum_k \theta_{j,k} l_{k,(i)}}}{\sum_i l_{k,(i)}} && \text{(apply first rewrite rule)}
\end{aligned}$$

Thus the second equation for NMF holds.

The importance of this lemma is in realising that the introduced set of equations are the maximum likelihood version of the mean-field algorithm for the Gamma-Poisson model. That is, NMF where $\boldsymbol{\Theta}$ is returned normalised is a maximum w.r.t. $\boldsymbol{\Theta}$ and $\boldsymbol{l}$ for the Gamma-Poisson likelihood $p(\boldsymbol{w} \mid \boldsymbol{\Theta}, \boldsymbol{l}, \boldsymbol{\alpha} = 0, \boldsymbol{\beta} = 0, K)$. This can be seen by inspection of the equations at the beginning of this section. Note, including a hidden variable such as $\boldsymbol{l}$ in the likelihood is not correct maximum likelihood, and for clustering yields the K-means algorithm.

5.2 Gibbs Sampling

There are two styles of Gibbs sampling that apply.

Direct Gibbs sampling: The basic Prichard's style Gibbs sampling works off Equation (6) or Equation (9) together with a Dirichlet prior for rows of $\mathbf{\Theta}$. The hidden variables can easily seen to be conditionally distributed in these equations according to standard probability distributions, in a manner allowing direct application of Gibbs sampling methods. The direct Gibbs algorithm for both versions is given in Figure 3. This Gibbs scheme then re-estimates each

1. For each document, retrieve the last $\boldsymbol{c}$ from store, then
   (a) Sample the hidden component variables:
      i. In the Gamma-Poisson case, $l_k \sim \text{Gamma}(c_k+\alpha_k, 1+\beta_k)$ independently.
      ii. In the Dirichlet-multinomial case, $\boldsymbol{l} \sim \text{Dirichlet}(\{c_k + \alpha_k \,:\, k\})$.
   (b) For each word j in the document with positive count w_j, the component counts vector, from Equation (5) and Equation (8),
   $$\{v_{j,k} \,:\, k = 1, \ldots, K\} \;\sim\; \text{Multinomial}\left(\left\{\frac{l_k\theta_{j,k}}{\sum_k l_k\theta_{j,k}} \,:\, k\right\}, w_j\right) .$$
   Alternatively, if the sequence-of-components version is to be used, the component for each word can be sampled in turn using the corresponding Bernoulli distribution.
   (c) Concurrently, record the log-probability
      i. In the Gamma-Poisson case, $p(\boldsymbol{w} \mid \boldsymbol{l}, \mathbf{\Theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}, K)$.
      ii. In the Dirichlet-multinomial case, $p(\boldsymbol{w} \mid \boldsymbol{m}, L, \mathbf{\Theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}, K)$.
   These represent a reasonable estimate of the likelihood $p(\boldsymbol{w} \mid \mathbf{\Theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}, K)$.
   (d) Concurrently, maintain the sufficient statistics for $\mathbf{\Theta}$, the total $\sum_i w_{j,(i)} n_{j,k,(i)}$ for each j, k over documents.
   (e) Store $\boldsymbol{c}$ for the next cycle and discard $\mathbf{v}$.
2. Using a Dirichlet prior for rows of $\mathbf{\Theta}$, and having accumulated all the counts $\mathbf{v}$ for each document in sufficient statistics for $\mathbf{\Theta}$, then its posterior has rows that are Dirichlet. Sample.
3. Report the total log-probability, and report.

Fig. 3. One Major Cycle of Gibbs Algorithm for DPCA

of the variables in turn. In implementation, this turns out to correspond to the mean field algorithms, excepting that sampling is done instead of maximisation or expectation.

Rao-Blackwellised Gibbs sampling: The Griffiths-Steyvers' style Gibbs sampling is a Rao-Blackwellisation (see [11]) of direct Gibbs sampling. It takes the likelihood versions of Equations (7) and (10) multiplied up for each document together with a prior on $\mathbf{\Theta}$, and marginalises out $\mathbf{\Theta}$. As before, denote the hidden variables $\mathbf{v}$ for the i-th document by adding an (i) subscript, as $v_{j,k,(i)}$. Then the Gamma-Poisson posterior for documents $i \in 1, \ldots, I$, with $\mathbf{\Theta}$ and $\boldsymbol{l}$ marginalised,

and constants dropped becomes:

$$\prod_i \left(\prod_k \frac{\Gamma(c_{k,(i)} + \alpha_k)}{(1+\beta_k)^{c_{k,(i)}+\alpha_k}} \prod_{j,k} \frac{1}{v_{j,k,(i)}!} \right) \prod_k \frac{\prod_j \Gamma\left(\gamma_j + \sum_i v_{j,k,(i)}\right)}{\Gamma\left(\sum_j \gamma_j + \sum_i c_{k,(i)}\right)} \tag{18}$$

A similar formula applies in the Dirichlet-multinomial version:

$$\prod_i \left(\prod_k \Gamma(c_{k,(i)} + \alpha_k) \prod_{jk} \frac{1}{v_{j,k,(i)}!} \right) \prod_k \frac{\prod_j \Gamma\left(\gamma_j + \sum_i v_{j,k,(i)}\right)}{\Gamma\left(\sum_j \gamma_j + \sum_i c_{k,(i)}\right)} \tag{19}$$

If sequence of components formulation is used, instead of bag of components, as discussed in Section 3.1, then the $v_{j,k,(i)}!$ terms are dropped. Now bag of components sampling would require a sampling scheme for each set $\{v_{j,k,(i)} : k \in 1,\ldots,K\}$. No such scheme appears to exist, other than doing sequence of components sampling, thus we do that. In one step, change the counts $\{v_{j,k,(i)} : k \in 1,\ldots,K\}$ by one (one is increased and one is decreased) keeping the total w_j constant. For instance, if a word has index j and is originally in component k_1 but re-sampled to k_2, then decrease $v_{j,k_1,(i)}$ by one and increase $v_{j,k_2,(i)}$ by one. Do this for every word in the document, thus it is done L times for the document per major sampling cycle.

This Rao-Blackwellised Gibbs algorithm for the Gamma-Poisson and Dirichlet-multinomial versions, both using sequence of components sampling, is given in Figure 3.

Implementation notes: Because Griffiths-Steyvers' scheme is a Rao-Blackwellisation of the direct Gibbs sampling, it implies that both $\boldsymbol{l}$ and $\boldsymbol{\Theta}$ are effectively re-estimated with each sampling step, instead of once after the full pass over documents. This is most effective during early stages, and explains the superiority of the method observed in practice. Moreover, it means only one storage slot for $\boldsymbol{\Theta}$ is needed (to store the sufficient statistics), whereas in direct Gibbs two slots are needed (current value plus the sufficient statistics). This represents a major savings in memory. Finally, $\boldsymbol{l}$ and $\boldsymbol{\Theta}$ can be sampled at any stage of this process (because their sufficient statistics make up the totals appearing in the formula), thus Gibbs estimates for them can be made as well during the MCMC process.

5.3 Other Aspects for Estimation and Use

A number of other algorithms are needed to put these models into regular use.

Component hyper-parameters: The treatment so far has assumed the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are given. It is more usual to estimate these parameters with the rest of the estimation tasks as done by [5, 9]. This is feasible because the parameters are shared across all the data, unlike the component vectors themselves.

1. Maintain the sufficient statistics for $\boldsymbol{\Theta}$, given by $\sum_i v_{j,k,(i)}$ for each j and k, and the individual component assignment for each word in each document.
2. For each document i, retrieve the component assignments for each word then
   (a) Recompute statistics for $\boldsymbol{l}_{(i)}$ given by $c_{k,(i)} = \sum_j v_{j,k,(i)}$ for each k from the individual component assignment for each word.
   (b) For each word in the document:
      i. Re-sample the component assignment for the word according to Equation (18) or (19) with the $v_{j,k,(i)}!$ terms dropped. If $v'_{j,k,(i)}$ is the word counts without this word, and $c'_{k,(i)}$ the corresponding component totals, then this simplifies dramatically (e.g., $\Gamma(x+1)/\Gamma(x) = x$).
         A. For the Gamma-Poisson, sample the component proportionally to
$$\frac{c'_{k,(i)} + \alpha_k}{1 + \beta_k} \frac{\gamma_j + \sum_i v'_{j,k,(i)}}{\sum_j \gamma_j + \sum_i c'_{k,(i)}}$$
         B. For the Dirichlet-multinomial, sample the component proportionally to
$$(c'_{k,(i)} + \alpha_k) \frac{\gamma_j + \sum_i v'_{j,k,(i)}}{\sum_j \gamma_j + \sum_i c'_{k,(i)}}$$
      ii. Concurrently, record the log-probability $p(\boldsymbol{w} \mid \boldsymbol{c}, \boldsymbol{\Theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}, K)$. It represents a reasonable estimate of the likelihood $p(\boldsymbol{w} \mid \boldsymbol{\Theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}, K)$.
      iii. Concurrently, update the statistics for $\boldsymbol{l}/\boldsymbol{m}$ and $\boldsymbol{\Theta}$.

Fig. 4. One Major Cycle of Rao-Blackwellised Gibbs Algorithm for DPCA

Estimating the number of components K: A simple scheme exists within importance sampling in Gibbs to estimate the evidence term for a DPCA model, proposed by [7], first proposed in the general sampling context by [10]. One would like to find the value of K with the highest posterior probability (or, for instance, cross validation score). In popular terms, this could be used to find the "right" number of components, though in practice and theory such a thing might not exist.

Use on new data: A typical use of the model requires performing inference related to a particular document. Suppose, for instance, one wished to estimate how well a snippet of text, a query, matches a document. Our document's components are summarised by the hidden variables $\boldsymbol{m}$ (or $\boldsymbol{l}$). If the new query is represented by $\boldsymbol{q}$, then $p(\boldsymbol{q}|\boldsymbol{m}, \boldsymbol{\Theta}, K)$ is the matching quantity one would like ideally. Since $\boldsymbol{m}$ is unknown, we must average over it. Various methods have been proposed [26, 7].

Alternative components: Hierarchical components have been suggested [7] as a way of organising an otherwise large flat component space. For instance, the Wikipedia with over half a million documents can easily support the discovery of several hundred components. Dirichlet processes have been developed as an alternative to the K-dimensional component priors in the Dirichlet-

multinomial/discrete model [32], although in implementation the effect is to use K-dimensional Dirichlets for a large K and delete low performing components.

6 Applications

This section briefly discusses a two applications of the methods.

6.1 Voting Data

One type of political science data are the *roll calls*. There were 459 roll calls in the US Senate in the year 2003. For each of those, the vote of every senator was recorded in three ways: 'Yea', 'Nay' and 'Not Voting'. The outcome of the roll call can be positive (e.g., Bill Passed, Nomination Confirmed) corresponding to 'Yea', or negative (e.g., Resolution Rejected, Veto Sustained). Hence, the outcome of the vote can be interpreted as the 101st senator, by associating positive outcomes with 'Yea' and negative outcomes with 'Nay'.

Application of the Method We can now map the roll call data to the DPCA framework. For each senator X we form two 'words', where $w_{X,y}$ implies that X voted 'Yea', and $w_{X,n}$ implies that X voted 'Nay'. Each roll call can be interpreted as a document containing a single occurrence of some of the available words. The pair of words $w_{X,y}, w_{X,n}$ s then treated as a binomial, so the multivariate formulation of Section 4.2 is used. Priors for $\mathbf{\Theta}$ were Jeffreys priors, $\boldsymbol{\alpha}$ was (0.1,0.1,...,0.1), and regular Gibbs sampling was used.

Special-purpose models are normally used for interpreting roll call data in political science, and they often postulate a model of rational decision making. Each senator is modelled as a position or an *ideal point* in a continuous spatial model of preferences [12]. For example, the first dimension often delineates the liberal-conservative preference, and the second region or social issues preference. The proximities between ideal points 'explain' the positive correlations between the senators' votes. The ideal points for each senator can be obtained either by optimization, for instance, with the optimal classification algorithm [28], or through Bayesian modelling [12].

Unlike the spatial models, the DPCA interprets the correlations between votes through membership of the senators in similar blocs. Blocs correspond to hidden component variables. Of course, we can speak only of the probability that a particular senator is a member of a particular bloc. The corresponding probability vector is normalized and thus assures that a senator is always a member of one bloc on the average. The outcome of the vote is also a member of several blocs, and we can interpret the membership as a measure of how influential a particular bloc is.

Visualization We can analyze two aspects of the DPCA model as applied to the roll call data: we can examine the membership of senators in blocs, and we can

examine the actions of blocs for individual issues. The approach to visualization is very similar, as we are visualizing a set of probability vectors. We can use the gray scale to mirror the probabilities ranging from 0 (white) to 1 (black).

As yet, we have not mentioned the choice of K - the number of blocs. Although the number of blocs can be a nuisance variable, such a model is distinctly more difficult to show than one for a fixed K. We obtain the following negative logarithms to the base 2 of the model's likelihood for $K = 4, 5, 6, 7, 10$: 9448.6406, 9245.8770, 9283.1475, 9277.0723, 9346.6973. We see that $K = 5$ is overwhelmingly selected over all others, with $K = 4$ being far worse. This means that with our model, we best describe the roll call votes with the existence of five blocs. Fewer blocs do not capture the nuances as well, while more blocs would not yield reliable probability estimates given such an amount of data. Still, those models are too valid to some extent. It is just that for a single visualization we pick the best individual one of them.

We will now illustrate the membership of senators in blocs. Each senator is represented with a vertical bar of 5 squares that indicate his or her membership in blocs. We have arranged the senators from left to right using the binary PCA approach of [14]. This ordering attempts to sort senators from the most extreme to the most moderate and to the most extreme again. Figure 5 shows the Democrat senators and Figure 6 the Republicans.

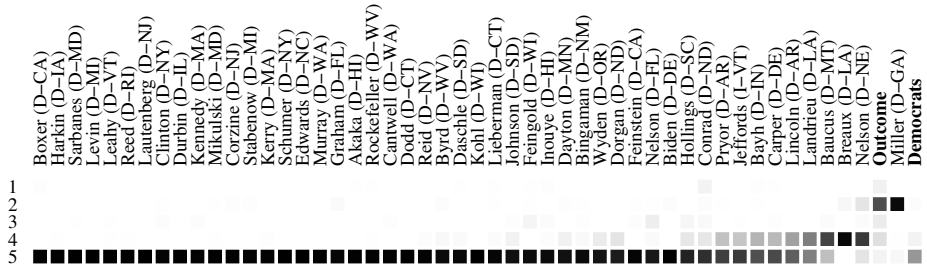

Fig. 5. Component membership for Democrats

We can observe that component 5 is the Democrat majority. It is the strongest overall component, yet quite uninfluential about the outcome. Component 4 is are the moderate Democrats, and they seem distinctly more influential than the Democrats of the majority. Component 3 is a small group of Republican moderates. Component 2 is the Republican majority, the most influential bloc. Component 5 is the Republican minority, not very influential. Component 5 tends to be slightly more extreme than component 4 on the average, but the two components clearly cannot be unambiguously sorted.

Fig. 6. Component membership for Republicans

7 Classification Experiments

We tested the use of MPCA in its role as a feature construction tool, a common use for PCA and ICA, and as a classification tool. For this, we used the 20 newsgroups collection described previously as well as the Reuters-21578 collection[6]. We employed the SVM^{light} V5.0 [22] classifier with default settings. For classification, we added the class as a distinct multinomial (cf. Section 4.2) for the training data and left it empty for the test data, and then predicted the class value. Note that for performance and accuracy, SVM is the clear winner: as the state of the art optimized discrimination-based system this is to be expected. It is interesting to see how MPCA compares.

Each component can be seen as generating a number of words in each document. This number of component-generated words plays the same role in classification as does the number of lexemes in the document in ordinary classification. In both cases, we employed the TF-IDF transformed word and component-generated word counts as feature values. Since SVM works with sparse data matrices, we assumed that a component is not present in a document if the number of words that would a component have generated is less than 0.01. The components alone do not yield a classification performance that would be competitive with SVM, as the label has no distinguished role in the fitting. However, we may add these component-words in the default bag of words, hoping that the conjunctions of words inherent to each component will help improve the classification performance.

For the Reuters collection, we used the ModApte split. For each of the 6 most frequent categories, we performed binary classification. Further results are disclosed in Table 2[7]. No major change was observed by adding 50 components to the original set of words. By performing classification on components alone, the results were inferior, even with a large number of components. In fact, with 300

[6] The Reuters-21578, Distribution 1.0 test collection is available from David D. Lewis' professional home page, currently: http://www.research.att.com/∼lewis

[7] The numbers are percentages, and 'P/R' indicates precision/recall.

components, the results were worse than with 200 components, probably because of over-fitting. Therefore, regardless of the number of components, the SVM performance with words cannot be reproduced by component-generated words in this collection. Classifying newsgroup articles into 20 categories proved more successful. We employed two replications of 5-fold cross validation, and we achieved the classification accuracy of 90.7% with 50 additional MPCA components, and 87.1% with SVM alone. Comparing the two confusion matrices, the most frequent mistakes caused by SVM+MPCA beyond those of SVM alone were predicting talk.politics.misc as sci.crypt (26 errors) and talk.religion.misc predicted as sci.electron (25 errors). On the other hand, the components helped better identify alt.atheism and talk.politics.misc, which were misclassified as talk.religion.misc (259 fewer errors) earlier. Also, talk.politics.misc and talk.religion.misc were not misclassified as talk.politics.gun (98 fewer errors). These 50 components were not very successful alone, resulting in 18.5% classification accuracy. By increasing the number of components to 100 and 300, the classification accuracy gradually increases to 25.0% and 34.3%. Therefore, many components are needed for general-purpose classification.

Table 2. SVM Classification Results

	SVM		SVM+MPCA	
CAT	ACC.	P/R	ACC.	P/R
earn	98.58	98.5/97.1	98.45	98.2/97.1
acq	95.54	97.2/81.9	95.60	97.2/82.2
moneyfx	96.79	79.2/55.3	96.73	77.5/55.9
grain	98.94	94.5/81.2	98.70	95.7/74.5
crude	97.91	89.0/72.5	97.82	88.7/70.9
trade	98.24	79.2/68.1	98.36	81.0/69.8

	MPCA (50 comp.)		MPCA (200 comp.)	
CAT	ACC.	P/R	ACC.	P/R
earn	96.94	96.1/94.6	97.06	96.3/94.8
acq	92.63	93.6/71.1	92.33	95.3/68.2
moneyfx	95.48	67.0/33.0	96.61	76.0/54.7
grain	96.21	67.1/31.5	97.18	77.5/53.0
crude	96.57	81.1/52.4	96.79	86.1/52.4
trade	97.82	81.4/49.1	97.91	78.3/56.0

From these experiments, we can conclude that components may help with tightly coupled categories that require conjunctions of words (20 newsgroups), but not with the keyword-identifiable categories (Reuters). Judging from the ideas in [21], the components help in two cases: a) when the co-appearance of two words is more informative than sum of informativeness of individual appearance

of either word, and b) when the appearance of one word implies the appearance of another word, which does not always appear in the document.

8 Conclusion

In this article, we have presented a unifying framework for various approaches to discrete component analysis, presenting them as a model closely related to ICA. We have shown the relationships between existing approaches here such as NMF, PLSI, LDA, MPCA and GaP, and presented the different algorithms available for two general cases, Gamma-Poisson and Dirichlet-multinomial. For instance NMF corresponds to a maximum likelihood solution for LDA. These methods share many similarities with both PCA and ICA, and are thus useful in a range of feature engineering tasks in machine learning and pattern recognition.

Acknowledgments

The first author's work was supported by the Academy of Finland under the PROSE Project, by Finnish Technology Development Center (TEKES) under the Search-Ina-Box Project, and by the EU 6th Framework Programme in the IST Priority under the ALVIS project. The results and experiments reported are only made possible by the extensive document processing environment and set of test collections at the COSCO group, and particularly the information retrieval software YDIN of Sami Perttu. The second author wishes to thank his advisor Ivan Bratko. The MPCA software used in the experiments was co-developed by a number of authors, reported at the code website[8].

References

[1] L. Azzopardi, M. Girolami, and K. van Risjbergen. Investigating the relationship between language model perplexity and IR precision-recall measures. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 369–370, 2003.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[3] J.M. Bernardo and A.F.M. Smith. *Bayesian Theory*. John Wiley, Chichester, 1994.

[4] E. Bingham, A. Kabán, and M. Girolami. Topic identification in dynamical text by complexity pursuit. *Neural Process. Lett.*, 17(1):69–83, 2003.

[5] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[6] W. Buntine. Variational extensions to EM and multinomial PCA. In *ECML 2002*, 2002.

[7] W. Buntine and A. Jakulin. Applying discrete PCA in data analysis. In *UAI-2004*, Banff, Canada, 2004.

[8] http://cosco.hiit.fi/search/MPCA

[8] W.L. Buntine, S. Perttu, and V. Tuulos. Using discrete PCA on web pages. In *Workshop on Statistical Approaches to Web Mining, SAWM'04*, 2004. At ECML 2004.

[9] J. Canny. GaP: a factor model for discrete data. In *SIGIR 2004*, pages 122–129, 2004.

[10] B.P. Carlin and S. Chib. Bayesian model choice via MCMC. *Journal of the Royal Statistical Society B*, 57:473–484, 1995.

[11] G. Casella and C.P. Robert. Rao-Blackewellization of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

[12] J. D. Clinton, S. Jackman, and D. Rivers. The statistical analysis of roll call voting: A unified approach. *American Political Science Review*, 98(2):355–370, 2004.

[13] M. Collins, S. Dasgupta, and R.E. Schapire. A generalization of principal component analysis to the exponential family. In *NIPS*13*, 2001.

[14] J. de Leeuw. Principal component analysis of binary data: Applications to roll-call-analysis. Technical Report 364, UCLA Department of Statistics, 2003.

[15] A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian Data Analysis*. Chapman & Hall, 1995.

[16] T.L. Griffiths and M. Steyvers. Finding scientific topics. *PNAS Colloquium*, 2004.

[17] T. Hofmann. Probabilistic latent semantic indexing. In *Research and Development in Information Retrieval*, pages 50–57, 1999.

[18] T. Hofmann and J.M. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):1–14, 1997.

[19] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

[20] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Netw.*, 13(4-5):411–430, 2000.

[21] A. Jakulin and I. Bratko. Analyzing attribute dependencies. In N. Lavrač, D. Gamberger, H. Blockeel, and L. Todorovski, editors, *PKDD 2003*, volume 2838 of *LNAI*, pages 229–240. Springer-Verlag, September 2003.

[22] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[23] D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[24] D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*12*, pages 556–562, 2000.

[25] P. McCullagh and J.A. Nelder. *Generalized Linear Models*. Chapman and Hall, London, second edition, 1989.

[26] T. Minka and J. Lafferty. Expectation-propagation for the generative aspect model. In *UAI-2002*, Edmonton, 2002.

[27] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *Proceedings of ACL-93*, June 1993.

[28] K.T. Poole. Non-parametric unfolding of binary choice data. *Political Analysis*, 8(3):211–232, 2000.

[29] J.K. Pritchard, M. Stephens, and P.J. Donnelly. Inference of population structure using multilocus genotype data. *Genetics*, 155:945–959, 2000.

[30] S.M. Ross. *Introduction to Probability Models*. Academic Press, fourth edition, 1989.

[31] M.A. Woodbury and K.G. Manton. A new procedure for analysis of medical classification. *Methods Inf Med*, 21:210–220, 1982.

[32] K. Yu, S. Yu, and V. Tresp. Dirichlet enhanced latent semantic analysis. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *Proc. of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005.