The Latent Bernoulli-Gauss Model for Data Analysis

Amnon Shashua
Gabi Pragier
School of Computer Science and Engineering
Hebrew University of Jerusalem

Abstract

We present a new latent-variable model employing a Gaussian mixture integrated with a feature selection procedure (the Bernoulli part of the model) which together form a "Latent Bernoulli-Gauss" distribution. The model is applied to MAP estimation, clustering, feature selection and collaborative filtering and fares favorably with the state-of-the-art latent-variable models.

1 Introduction

We present a new mixture model for collections of discrete data with applications to clustering through MAP classification, supervised learning, feature selection and collaborative filtering. In the language of text modeling, the algorithm integrates modeling of word frequencies with a feature selection procedure into a single latent class distribution model. The algorithm defines two types of words (i) keywords representing "important" words associated with high frequency appearance, and (ii) all remaining words (not including stop-words which are omitted from consideration). All keywords are "topic specific" modeled by a mixture of Gaussians (one per topic) and all remaining words are considered "topic unspecific" are modeled by a single Gaussian. The decision of which are the keywords of a document is modeled by a latent Bernoulli process — thus together we have a "Latent Bernoulli-Gauss" (LBG) model.

We present the LBG model in sec. 2 and its applications in sec. 2.3. In sec. 3 we present a detailed discussion of the merits of LBG as compared to existing latent-variable models including Mixture-of-Unigrams (MOU) (Nigam et al., 2000), probabilistic Latent Semantic Indexing (pLSI) (Hofmann, 1999) and Latent Dirichlet Allocation (LDA) (Blei et al., 2003). We conducted a series of experiments on public datasets covering a spectrum of information retrieval applications — a detailed discussion of experimental results and comparisons to MOU, LDA and pLSI is in sec. 4.

We use the language of text collections throughout the paper, referring to measurements as "word frequencies" and "documents". Nevertheless, the LBG model is general and can be applied (and is applied in sec. 4) to a wide range of data analysis tasks.

2 The Bernoulli-Gauss Mixture Model

Consider a code-book of size $n$ representing the vocabulary of $n$ words in a dictionary. A document is an unordered collection of $N$ words $w_1, \ldots, w_N$ where $w_i \in \{1, \ldots, n\}$. A document $d$ is represented by the $n$ frequencies of word appearances normalized in a proper manner (in text applications we use the term-frequency-inverse-document-frequency (tf-idf) normalization), resulting in $d = (m_1, \ldots, m_n)$ a set of non-negative real numbers.

For a document $d$, we distinguish between a "keyword" which is associated with a high frequency and other low-frequency words of the document. A keyword is another way of saying that the word is "important" for that document. Let $x \in \{0, 1\}^n$ be an indicator set where $x_i = 1$ if the $i$'th word in the code-book is a keyword and $x_i = 0$ otherwise. We assume that the keywords are modeled by a topic-specific Normal distribution whereas all other words are modeled by a single Gaussian. The decision of which are the keywords of a document is modeled by a latent Bernoulli process — thus together we have a "Latent Bernoulli-Gauss" (LBG) model.

We present the LBG model in sec. 2 and its applications in sec. 2.3. In sec. 3 we present a detailed discussion of the merits of LBG as compared to existing latent-variable models including Mixture-of-Unigrams (MOU) (Nigam et al., 2000), probabilistic Latent Semantic Indexing (pLSI) (Hofmann, 1999) and Latent Dirichlet Allocation (LDA) (Blei et al., 2003). We conducted a series of experiments on public datasets covering a spectrum of information retrieval applications — a detailed discussion of experimental results and comparisons to MOU, LDA and pLSI is in sec. 4.

We use the language of text collections throughout the paper, referring to measurements as "word frequencies" and "documents". Nevertheless, the LBG model is general and can be applied (and is applied in sec. 4) to a wide range of data analysis tasks.
evaluated at \( z \), and \( \theta = (p, c, \sigma) \) holds the parameters of the model. If the \( i \)th code word is a keyword \((x_i = 1)\) then the word’s frequency \( m_i \) is governed by a topic-specific Gaussian distribution \( N(c_i, \sigma_i^2) \), otherwise \( m_i \sim N(c_i, \sigma_i^2) \) a topic-unspecific Gaussian distribution which we refer to as a ”cross Gaussian”. The probability \( Pr(d \mid \theta) \) of a document \( d \) to be generated by the LBG model is found by the mixture:

\[
Pr(d \mid \theta) = \sum_{s=1}^{k} \lambda_s Pr(d \mid y = s, \theta)Pr(y = s \mid \theta)
\]

where \( \sum_s \lambda_s = 1 \). Given a training set of documents \( \mathcal{D} = (d_1, ..., d_m) \) we wish to fit the model parameters \( \theta, \lambda \) and select the important code words for each document, i.e., estimate \( \mathcal{X} = (x_1, ..., x_m) \) where \( x_j \in \{0, 1\}^n \) is the keyword indicator set associated with \( d_j \). We alternate between two procedures: (i) Maximum-Likelihood (ML) estimation of \( \{\theta, \lambda\} \) given \( \mathcal{X} \) and, (ii) a procedure for estimating \( \mathcal{X} \) given \( \{\theta, \lambda\} \).

The ML estimation of \( \{\theta, \lambda\} \) given an i.i.d. training set \( \{\mathcal{D}, \mathcal{X}\} \) takes the form:

\[
\max_{\theta, \lambda} \sum_{j=1}^{m} \log Pr(d_j \mid \theta, x_j) = \max_{\theta, \lambda} \sum_{j=1}^{m} \left( \log \left( \sum_{s=1}^{k} \lambda_s Pr(d_j \mid y_j = s, \theta, x_j) \right) \right)
\]

where \( Pr(d_j \mid y_j = s, \theta, x_j) \) is given by:

\[
\prod_{i=1}^{n} (p_{si} N(m_{ji}; c_i, \sigma_i^2))^{x_{ji}} ((1 - p_{si}) N(m_{ji}; c_i, \sigma_i^2))^{1-x_{ji}}.
\]

Using the Expectation-Maximization (EM) iterative update (Dempster et al., 1977), the following auxiliary function is optimized during the M-step:

\[
\max_{\theta, \lambda} \sum_{j=1}^{m} \sum_{s=1}^{k} \mu_{sj}^{(t)} \log (\lambda_s Pr(d_j \mid y_j = s, \theta, x_j))
\]

where \( \mu_{sj}^{(t)} = Pr(y_j = s \mid d_j, x_j, \theta^{(t)}) \) is the posterior probability given the parameters at iteration \( (t) \). Optimizing over the auxiliary function at step \( (t) \) introduces an update rule for \( \theta, \lambda \):

\[
\lambda_s \leftarrow \frac{1}{m} \sum_{j=1}^{m} \mu_{sj}^{(t)}, \quad p_{si} \leftarrow \frac{1}{\sum_{j=1}^{m} \mu_{sj}^{(t)}} \sum_{j=1}^{m} \mu_{sj}^{(t)} x_{ji} \quad (2)
\]

\[
c_{si} \leftarrow \frac{1}{\sum_{j=1}^{m} \mu_{sj}^{(t)} x_{ji}} \sum_{j=1}^{m} \mu_{sj}^{(t)} x_{ji} m_{ji} \quad (3)
\]

\[
\sigma_{si}^2 \leftarrow \frac{1}{\sum_{j=1}^{m} \mu_{sj}^{(t)} x_{ji}} \sum_{j=1}^{m} \mu_{sj}^{(t)} x_{ji} (m_{ji} - c_{si})^2 \quad (4)
\]

The parameters \( c_i, \sigma_i^2 \) of the cross-Gaussians are estimated directly from \( \mathcal{D}, \mathcal{X} \) since they do not depend on the choice of topics. The posteriors are updated during the E-step via application of the Bayes rule:

\[
\mu_{sj}^{(t+1)} \propto \lambda_s^{(t)} Pr(d_j \mid y_j = s, \theta, \theta^{(t)}), \quad (5)
\]

where \( \propto \) stands for equality up to normalization, i.e., \( \sum_s \mu_{sj} = 1 \).

The estimation of \( \mathcal{X} \) given the data \( \mathcal{D} \) and the current estimation of parameters \( \{\theta, \lambda\} \) is based on the following analysis. Consider Natural numbers \( q_s \in \mathcal{N}, s = 1, ..., k \), representing the number of important code words associated with topic \( s \). The expected number of important code words \( g_j \) in document \( d_j \) is given below:

\[
g_j = \sum_{i=1}^{n} x_{ji} = \sum_{s=1}^{k} \mu_{sj} q_s. \quad (6)
\]

In other words, the indicator set \( \mathcal{X} \) is fully determined by \( q_1, ..., q_k \) and the posteriors \( \mu_{sj} \) (which are estimated during the EM step above). The indicator \( x_j \) for document \( d_j \), for instance, is defined by the top \( g_j \) highest frequency code words. Our task, therefore, is to derive a procedure for estimating \( q_1, ..., q_k \) given the parameters \( \theta, \lambda \) and \( \mu \) estimated during the EM steps.

We will begin by establishing an algebraic constraint between \( q = (q_1, ..., q_k) \) and the parameters \( p, \mu, \pi \):

**Claim 1** Let \( b = (b_1, ..., b_k) \) defined by \( b_s = \sum_i \mu_{si} / \sum_i p_{si} \) for \( s = 1, ..., k \) and let \( U \) be an \( k \times m \) matrix holding the posteriors, \( U_{sj} = \mu_{sj} \). Then,

\[
b = UU^T q \quad (7)
\]

**Proof:** consider the formula representing the expected number of important words for a document of topic \( s \):

\[
E_s = 1 / \sum_j \mu_{sj} \sum_{j=1}^{m} \mu_{sj} g_j.
\]

On the other hand, clearly, \( E_s = \sum_i p_{si} \) since \( p_{si} \) is the probability that the \( i \)th code word is important for documents of topic \( s \). Substituting the definition of \( g_j \) from eqn. 6, we obtain:

\[
(\sum_s \mu_{sj})(\sum_s p_{si}) = \sum_{j=1}^{m} \mu_{sj} \sum_{r=1}^{k} \mu_{sj} q_r,
\]

where the right hand side is the \( s \)th coordinate of \( UU^T q \). [ ]

The conditional-independence assumption \( w_i \perp w_j \mid y \) (Naive-Bayes) creates "over-confident" posteriors, i.e., \( \mu_{sj} \rightarrow \{0, 1\} \) — a well-known by-product (or side-effect) of the Naive Bayes assumption (see Domingos & Pazzani, 1997) for a discussion). As a result,
the constraint $UU^T q = b$ is simplified considerably: $UU^T \approx \text{diag}(\delta_1, \ldots, \delta_k)$, where $\delta_s \approx \sum_j \mu_{sj}^2 \approx \sum_j \mu_{sj}$. Eqn. 7, therefore, reduces to:

$$\sum_{i=1}^{n} p_{si} = q_s,$$  \hspace{1cm} (8)

for $s = 1, \ldots, k$. Eqn. 8 is not an effective update rule for setting $q_1, \ldots, q_k$ because (i) there is no built-in drive to generate a sparse $p$, which as a result, a large number of small-valued entries in $p$ will inflate the value of $q_s$, and (ii) once entries of $\mu$ settle on $\{0, 1\}$ values, the indicator set $X$ will remain fixed.

A more effective use of Eqn. 8 is to set $q_s$ as the top number of entries in $p$:

$$q_s^{(t+1)} = \left| \left\{ i : p_{si} \geq T_s^{(t)} \right\} \right|,$$

for some, iteration dependent, threshold $T_s$. In the following section we use a similar analysis to derive the value of $T_s$ which will conclude the Bernoulli-Gauss mixture algorithm.

### 2.1 Update Rule for $q_1, \ldots, q_k$

Let $q_s^*$ be the (unknown) ground truth value for $q_s$. Since $g_j$ (eqn. 6) is the number of keywords in document $d_j$, the probability that a keyword will be selected in $d_j$, conditioned by topic $s$, is $\min\{g_j/q_s^*, 1\}$. The probability that a keyword will be selected in $d_j$ and the topic is $s$ is a random variable with a Bernoulli distribution with the probability of "success": $\mu_{sj} \min\{g_j/q_s^*, 1\}$. The expected number of times a keyword is selected over the corpus of $m$ documents of topic $s$ is the sum of expectations of $m$ Bernoulli trials:

$$\sum_{j=1}^{m} \mu_{sj} \min\{g_j/q_s^*, 1\} = q_s^*.$$

On the other hand, the expected number of times the $i$'th code-word (not necessarily a keyword) is selected in documents of topic $s$ is: $m\lambda_s p_{si}$. As a result, for the $i$'th code-word to be a keyword a document of topic $s$ the following condition must be satisfied:

$$m\lambda_s p_{si} \geq \sum_{j=1}^{m} \mu_{sj} \min\{g_j/q_s^*, 1\} \geq \frac{1}{n} \sum_{j=1}^{m} \mu_{sj} g_j,$$

where the first inequality is due to the rhs being a lower bound for a word to be a keyword, and the latter inequality is due to $1 \leq q_s^* \leq n$. After rearranging terms and substituting eqn. 6 for $g_j$ we obtain:

$$q_s^{(t+1)} = \left| \left\{ i : p_{si} \geq \frac{1}{n} \sum_{j=1}^{m} \mu_{sj} \sum_{r=1}^{k} \mu_{sj} q_s^{(t)} \right\} \right|.$$  \hspace{1cm} (9)

Note that the right-hand side is the $s$'th coordinate of $UU^T q$ scaled by $1/(n \sum_j \mu_{sj})$. Given that the posteriors $\mu_{sj}$ approach $\{0, 1\}$ values, the condition above reduces to:

$$q_s^{(t+1)} = \left| \left\{ i : p_{si}^{(t+1)} \geq \frac{1}{n} q_s^{(t)} \right\} \right|.$$  \hspace{1cm} (9)

To conclude, the Bernoulli-Gauss mixture algorithm is summarized in Alg. 1. The stopping criteria is when Claim 1 is satisfied, but in practice it is sufficient to satisfy its reduced form eqn. 9.

### 2.2 Evaluating the Model on Novel Documents

Given a new document $d = (m_1, \ldots, m_n)$, where $m_i$ is the frequency (tf-idf) of the $i$'th code-word in the document, we wish to evaluate the probability $Pr(d)$ of $d$ to arise from the model, and the posteriors $\mu_{si} = Pr(d \mid y = s, x_s)$ which provide classification (topic assignment) information. A necessary ingredient in those calculations is the estimation of the keyword indicator set $x \in \{0, 1\}^n$ for document $d$. To estimate $x$ associated with the novel document $d$ we perform the following steps:

1. For $s = 1, \ldots, k$: (i) define $x_s$ as the indicator set defined by the top $q_s$ code words in $d$, (ii) compute $\hat{\mu}_s \propto \lambda_s Pr(d \mid y = s, x_s)$ where $Pr(d \mid y = s, x_s)$ is defined in eqn. 1.

2. Set $x$ as the top $(1/\sum_s \hat{\mu}_s) \sum_s \hat{\mu}_s q_s$ code words in $d$.

Once $x$ is estimated, one can readily compute the posterior $\mu_s \propto \lambda_s Pr(d \mid y = s, x, \theta)$, $s = 1, \ldots, k$, and $Pr(d)$ from: $Pr(d) = \sum_{s=1}^{k} \lambda_s Pr(d \mid y = s, x, \theta)$.

### 2.3 Applications of the Model

The Bernoulli-Gauss mixture model can be used in a number of ways and for different data analysis applications, as described below:

**Clustering:** given documents $d_1, \ldots, d_m$, cluster them into $k$ classes. Moreover, given a novel document $d$ determine its class association. The posteriors $\mu_{sj}$ for $d_j$ and class $s$ provide the class assignment of document $d_j$. Since posteriors are "over-confident" due to the Naive Bayes assumption, the assignment is "hard" in practice. For a new document $d$, the posteriors $\mu_s$ (see Sec. 2.2) provide the class assignments for $s = 1, \ldots, k$.

**Supervised Inference:** given a training set of documents with class labels in the set $\{1, \ldots, k\}$ we wish to determine the class membership of a given novel document. A possible approach is to estimate a LBG model separately for each class producing the model param-
Algorithm 1 Bernoulli-Gauss Mixture

Input: Given a training set of documents $D = (d_1, \ldots, d_m)$ we wish to fit the model parameters $\lambda$ and $\theta = ( \mathbf{p}, \mathbf{c}, \mathbf{\sigma})$ for $k$ topics and the Natural numbers $q_1, \ldots, q_k$ of top ranking (by tf-idf) words per topic.

Initialization: Set initial values $\lambda^{(0)}, \theta^{(0)}, q^{(0)}$. Set the indicators $\chi^{(0)}$ from $D$ and $q^{(0)}$, i.e., $x_{ji} = 1$ if the if-idf value $m_{ji}$ is among the top $(1/k) \sum_s q_s^{(0)}$ entries in $d_j$. Set $t = 0$.

repeat

$t \leftarrow t + 1$
Update the posteriors $p_{s|j}^{(t)}$ according to Eqn. 5 for $j = 1, \ldots, m$ and $s = 1, \ldots, k$.
Update $\lambda^{(t)}, \mathbf{p}^{(t)}, \mathbf{c}^{(t)}, \mathbf{\sigma}^{(t)}$ using Eqs. 2-4 and then update the cross-Gaussians.
Set $q^{(t)}$ using eqn. 9.
Set $\chi^{(t)}$: $x_{ji} = 1$ if the if-idf value $m_{ji}$ is among the top $\sum_s \mu_{s|j}^{(t)} q_s^{(t)}$ entries in $d_j$, for $i = 1, \ldots, n$ and $j = 1, \ldots, m$.

until $\sum_{s=1}^{k} \left( q_s^{(t)} - \sum_i p_{s|i}^{(t)} \right)^2 < \epsilon$

eters $\lambda_l, \theta_l, q_l, l = 1, \ldots, h$, and then choose the class with the highest probability: $\text{argmax}_i Pr(d \mid \lambda_l, \theta_l, q_l)$.

Feature Selection: we can use the Bernoulli-Gauss mixture model for selecting features. The selection criteria is based on $p_{s|i}$ which is the probability that the $i$'th code word (feature) is a keyword for topic $s$. We "de-select" a feature $i$ if $p_{s|i} < \delta$ for some threshold $\delta$ for all $s = 1, \ldots, k$, i.e., a feature that is not a keyword in all topics is removed from the set of selected features. In Sec. 4 we apply the feature selection scheme above as a filter for Support-Vector-Machine (SVM) classification and for K-means clustering.

Collaborative Filtering: there are applications where the indicator set $x \in \{0,1\}^n$ is known, and moreover when $x_i = 0$ the frequency of the $i$'th code word $m_i$ is unknown. Collaborative Filtering (CF) is an example of this class of applications where $d = (m_1, \ldots, m_n)$ is a list of discrete movie ratings with $m_i \in \{1, \ldots, 5\}$ (stars), of an individual. Each individual rates some of the movies, thus $x_i = 1$ for movies being rated and $x_i = 0$ otherwise. Given a subset of ratings made by a new individual, the task of CF is to predict movie ratings which were not part of the original subset.

In this case, the cross-Gaussians are dropped from the model, i.e.,

$$Pr(d \mid y = s) = \prod_{i=1}^{n} \left( p_{s|i} N(m_i; c_{s|i}, \sigma_{s|i}^2) \right)^{x_i} (1 - p_{s|i})^{1-x_i}. \tag{10}$$

From the training ratings $\{d_j, x_j\}, j = 1, \ldots, m$, we estimate the model parameters $\theta, \lambda$ using Eqs. 2-4 (there is no need to estimate $q_1, \ldots, q_k$ since the indicator sets are known). We are given a new rating $\{d, x\}$ where $d = (m_1, \ldots, m_n)$ and $x_i = 1$ when $m_i > 0$. Let $i \in \{1, \ldots, n\}$ be a movie we wish to predict its rating by the individual $d$. Similarly to the "Forced Prediction" protocol (Breese et al., 1998), we wish to estimate the probability $Pr(m_i = t \mid d, \theta)$ for $t = 1, \ldots, 5$. We start by setting $x_i = 1$ (originally it was zero):

$$Pr(m_i = t \mid d) = \sum_{s=1}^{k} Pr(m_i = t \mid y = s) Pr(y = s \mid d),$$

where

$$Pr(m_i = t \mid y = s) = N(t; c_{s|i}, \sigma_{s|i}^2),$$

and the posterior $Pr(y = s \mid d) \propto \lambda_s Pr(d \mid y = s)$ is estimated through eqn. 10. The movie rating prediction $t^*$ is found by: $t^* = \text{argmax}_t Pr(m_i = t \mid d)$.

3 Relationship with Other Latent Variable Models

On a simplistic level, the Bernoulli-Gauss mixture model can be viewed as a Gaussian mixture model integrated with a feature selection procedure (the Bernoulli part of the model). On a deeper level, however, there are subtleties that have to do with the positioning of LBG with respect to MOU, LDA and pLSI and specifically the manner in which LBG is a generative model like MOU and LDA, which we will describe below.

One difference is that LBG models the frequency of a code word (per topic) as a Gaussian whereas MOU, LDA and pLSI model the probability of appearance of code-words as a multinomial — which at the limit are really the same, as described next. Let $\beta_{s|i} = Pr(w = i \mid y = s)$ be the probability of drawing the $i$'th code-word given the $s$'th topic. The number of appearances $m_i$ of the $i$'th code-word in a document is governed by a Binomial distribution $m_i \sim \text{Bin}(N, \beta_{s|i})$ where $N$ is the number of words in the document. By the De-Moivre-Laplace theorem,
as $N \to \infty$, $m_i \sim \mathcal{N}(N \beta_{si}, N \beta_{si}(1 - \beta_{si}))$. Therefore, in practice since the number of words $N$ is a document is typically large, the estimated means $c_{si}$ in the LBG model are equal to $N \beta_{si}$ in the multinomial models.

The De-Moivre-Laplace argument above is also relevant for the justification of a Gaussian distribution as a model of word frequencies (or any other non-negative data). It implies that the probability of a negative value (in the generative sense) is vanishingly small. Successful attempts in using Gaussian mixtures in non-negative numerical contexts, such as for collaborative filtering, include (Hofmann, 2003). In practice we have not observed any problematic issue with a Gaussian modeling and our experimental reports across a number of application domains (text analysis included) make that point as well.

It will be convenient, in this section, to represent a document $d = (w_1, ..., w_N)$ by the (unordered) set of words $w_i \in \{1, ..., n\}$ taking values from a vocabulary of $n$ code-words. We will begin the discussion with the comparison between the MOU model and LBG. A document is generated by the MOU model by a draw from a mixture of multinomials as follows. A topic is drawn by tossing a $k$-faced die whose faces have probabilities $\lambda_s = Pr(y = s)$. A word is drawn by the toss of an $n$-faced die where we have $k$ such dice each representing a topic $s = 1, ..., k$, with $\beta_{si}$ (as defined above) representing the probability of the $i$th face of the $n$-face word-die associated with topic $s$. The $N$ words of a document are generated by (i) draw a topic $s$ by tossing the $k$-faced topic-die, then repeat $N$ times; (ii) draw a code-word by tossing the $s$th word-die. In formal language,

$$Pr(d) = \prod_{s=1}^{k} \lambda_s \prod_{i=1}^{n} \beta_{si}^{m_i}.$$ 

The model parameters $\lambda, \beta$ can be estimated by the EM algorithm. The MOU model is simple and very popular in text analysis circles. However, it has a number of drawbacks which have served as a catalyst for introducing new algorithms, notably pLSI and LDA. The notion that all code-words appearance is governed by the choice of a single topic is too simplistic. First, there are code-words which have a low probability of appearance in all topics, i.e., are essentially topic-independent, yet are not stop-words. These words undergo "starvation" in the MOU model as they almost never have a chance to be appear in a document generated by MOU. Second, polysemy — the coexistence of multiple meanings for a code-word — is not modeled by MOU. Consider a document $d$ and a code-word $w$. In MOU the posterior probability $Pr(y = s \mid w, d)$ is independent of $d$:

$$Pr(y = s \mid w, d) \propto Pr(w \mid y = s)Pr(y = s),$$ 

therefore it is not possible to convey multiple meanings for the code-word $w$ as a function of other words in the document $d$.

In the LBG model, the single topic assumption applies only to a selected set of code-words, whereas all other code-words are governed by a topic-unspecific distribution. The manner in which this principle plays in a generative model is described formally as follows:

$$Pr(d, x) = \sum_{s=1}^{k} \lambda_s \prod_{r=1}^{N} Pr(w_r, x_r \mid y = s),$$

where

$$Pr(w_r = i, x_r \mid y = s) \propto \begin{cases} p_{si} \frac{1}{N} c_{si} & \text{if } x_i = 1 \\ (1 - p_{si}) \frac{1}{N} c_i & \text{if } x_i = 0 \end{cases}$$

In other words, the $N$ words of a document $d$ are generated through the following steps:

- Draw a topic $s$ by tossing the $k$-faced topic-die.
- Toss $n$ coins with biases $p_{si}$, $i = 1, ..., n$ to draw the indicator vector $x \in \{0, 1\}^n$.
- Create a $n$-faced word-die by setting $\beta_{si}$ to $(1/N)c_{si}$ if $x_i = 1$ or to $(1/N)c_{i}$ if $x_i = 0$. The probability $\beta_{si}$ of the $i$th face of the word-die is $(1/Z)\beta_{si}$ where $Z$ is a normalization factor such that $\sum_i \beta_{si} = 1$.
- Repeat $N$ times: draw a word from the word-die constructed above.

In other words, in the LBG model the word-die is generated per document not only on the basis of the topic selection but also based on the selection of keywords. With regard to polysemy, the posterior probability $Pr(y = s \mid w, d)$ now depends on $d$:

$$Pr(y = s \mid w, d) \propto Pr(w \mid y = s, d)Pr(y = s),$$

unlike MOU. The LBG model therefore addresses the two main drawbacks of MOU: first being that the single-topic assumption does not apply to the entire document, but only to selected keywords; and secondly, that the word generation process depends on the document thereby allowing multiple meanings to words. Both of those "upgrades" make the underlying model assumptions more realistic than MOU. Consequently, the LBG model can be considered as a natural extension of the MOU model where some of the limiting (and unrealistic) assumptions of MOU are relaxed.

The LDA model addresses the single-topic assumption of MOU by allowing multiple topics per document in
the following manner. To generate the $N$ words of a document $d$, (i) a $k$-faced topic-die is generated by sampling from a Dirichlet distribution with parameters $\alpha, \ldots, \alpha_k$, then (ii) repeat $N$ times: (a) sample a topic $s$ by tossing the topic-die, and (b) sample a word by tossing the word-die $\beta_s$.

The parameters $\alpha, \beta$ of the LDA model are learned through a Variational EM algorithm. Unlike MOU and LBG, in the LDA model the topic is selected per word rather than once per document. This approach definitely solves the single-topic limitation of MOU and also the polysemy issue since the posterior $Pr(y = s \mid w, d)$ depends on $d$:

$$ Pr(y = s \mid w, d) \propto Pr(w \mid y = s)Pr(y = s \mid d). $$

However, there is a price to pay for the powerful generality of the LDA model. First, the posteriors $P(y = s \mid d)$ are computationally intractable and instead are replaced by a mean-field ”surrogate” approximation or by sampling methods. Secondly, by design, LDA requires a relatively large number of topics $k$ (around $\sim 50$) which is fine in the world of text but is limiting to other data analysis domains where the number of ”topics” are known to be small (like clustering applications).

In practice, LDA is often used for dimensionality reduction (using the variational parameters $\gamma \in \mathbb{R}^k$ per document) as a filter for SVM classification and for supervised classification by performing a separate LDA modeling per class. Despite the reservations above, there are situations where the powerful generality of the LDA model pays off — in the domain of text this happens when two topics are very similar. In such cases, the modeling capacity of MOU and LBG is too limited and cannot separate the two classes (see sec. 4 for details).

The pLSI model represents the training data as a mixture of multinomials and, like LDA, also allows for multiple topics per document. The pLSI model (unlike MOU, LDA and LBG) is not generative, i.e., there is no natural way to use the model to assign probability to a novel document. Related to that, the number of parameters of the model grows linearly with the training set thus risking an over-fitting phenomenon to occur. The pLSI model, therefore, is not a natural candidate for classification tasks because a novel data instance cannot be classified without essentially retraining the entire dataset. We refer the reader to (Blei et al., 2003) for a detailed comparison between LDA and pLSI. We have included pLSI in our experiments (sec. 4) as one can often obtain good performance if retraining is allowed during classification of a novel document.

4 Experiments

We conducted experiments with MAP classification, feature selection as a filter for SVM and K-means, supervised classification fitting a model per class and collaborative filtering. Those experiments were conducted on a number of datasets including 20Newsgroup\(^1\), 100KMovieLens, and Spambase from UCI ML repository.

MAP Unsupervised Classification: we begin with an unsupervised classification experiment using the MAP output of our model (the posteriors $\mu_s = Pr(y = s \mid d)$). We randomly split the data set into training and test subsets, generated by mixing records from all the topics in the data set. Having stripped all the record headers, a code-book is created, comprising of all words which are not stop-words in the data set. We trained a MAP classifier with our model and evaluated the classification output by comparing the cluster label of each record with its true label, as per the 20NewsGroup data set.

In order to measure the clustering performance, we use the zero-one loss function, as follows. Given the $i$’th posting, let $s_i$ and $k_i$ be the obtained cluster label and the true label, respectively. The accuracy (AC), is defined by $AC = (1/m) \sum_{i=1}^{m} \delta(k_i, map(s_i))$, where $\delta(x, y)$ is an indicator function that equals one if $x = y$ and zero otherwise; and $map(s_i)$ is the permutation mapping function that maps each cluster label $s_i$ to its equivalent label from the data set. The optimal mapping is obtained by the Kuhn-Munkres algorithm (Lovasz & Plummer, 1986). We compared our results with those of MOU, pLSI and LDA. For the latter, a clustering decision was made by examining the $\phi_i$ variational parameters that are introduced for each record. We repeated the experiments several times and the average results are reported in Table 1.

Note that pLSI retrained the entire data for each new test record thus skewing the comparison — yet it is interesting to note that LBG matched the performance nevertheless. Note the large performance gap between LBG and MOU underscoring the significant upgrade to the MOU model. We conjecture that the relatively low accuracy obtained by the LDA model is related to the mean-field approximation and as mentioned above, LDA is hardly ever used for MAP applications for presumably the same reasons.

Feature Selection: we compared the performance of our feature selection procedure, as described in sec. 2.3, with the dimensionally reduction offered by

\(^1\)The 20NewsGroup data set, taken from the Usenet Newsgroup Collection, consists of some 20,000 newsgroup postings, each one categorized to a different topic where each topic contains 1000 documents.
Figure 1: 20NewsGroup classification result on a binary classification problem, using SVM on the reduced set of features. Graph (a) is misc.forsale vs. rec.sport.baseball. Graph (b) is comp.graphics vs. comp.os.ms_windows.misc.

Table 1: MAP classification performance comparison for the 20NewsGroup data set.

|       | LBG | MOU | pLSI | LDA |
|-------|-----|-----|------|-----|
| 28%   | 15% | 27% | 12%  |

In the second experiment, we performed an unsupervised classification using K-means clustering on the filtered representations and without the filter (the raw data). Results for both semantically-close and semantically-unrelated pairs of classes are shown in Tables 2 and 3. One can see that LDA can produce a superior accuracy when the two classes are semantically-close (comp.os.ms_windows.misc versus comp.graphics). LBG on the other hand consistently outperformed LDA for semantically-unrelated classes.

Collaborative Filtering: We used the 100KMovieLens Collaborative-Filtering data, which consists of approximately 100,000 ratings for 1,682 movies by 943 viewers. As discussed in sec. 2.3, we train our model using a fully-observed set of viewers. Then, for every test viewer, we suppress a single, randomly-chosen movie rating. Our task is to predict the rating, given all the other movies for which that viewer has voted (known as the "Forced Prediction" protocol). Adopting Hofmann (2003) and Breese (1998), we use two evaluation metrics which measure the distance of the estimated vote \( \hat{m} \) from the true vote \( m \) — the mean absolute error MAE, \( \text{avg}(|\hat{m} - m|) \), and the rooted mean squared error RMSE \( \text{avg}((\hat{m} - m)^2) \). We then compared our method to Gaussian-pLSA proposed by (Hofmann, 2003) and to the Baseline method that simply outputs the mean vote over the entire training data for each movie. The results are displayed in Table 5. Note that LDA and pLSI do not naturally accommodate the Forced Prediction protocol as they do not measure word frequencies, thus were omitted from the comparison. One can see that LBG produced a lower MAE error compared to both Gaussian-pLSA and the Baseline method and slightly lower error on the RMSE measure (compared to Baseline).

Spam Filtering: The Spambase data set from the UCI Machine Learning Repository dataset consists of 4601 of emails ("documents"), characterized by 54 attributes ("words") plus a class label ("spam" = positive/"ham" = negative) where 39% of the emails are labeled as spam. We begin with an unsupervised MAP estimation where Table 4 displays the performance of LBG against MOU and pLSI (where with pLSI a retraining is required for each test data). Note the performance gap between LBG and pLSI —

---

"LDA at http://chasen.org/~daiti-m/dist/lda/"
Table 2: K-means classification for semantically-close classes.

|                | comp.os.ms.windows.misc | talk.politics.mideast | rec.sport.baseball | talk.religion.misc |
|----------------|-------------------------|-----------------------|--------------------|-------------------|
|                | comp.graphics           | talk.politics.misc    | rec.sport.hockey   | talk.religion.cristianity |
| LBG            | 63.25%                  | 72.75%                | 52.875%            | 57.625%           |
| LDA            | 77.5%                   | 57.75%                | 53%                | 58.375%           |
| All            | 50.25%                  | 58.625%               | 50.375%            | 50.5%             |

Table 3: K-means classification for semantically-unrelated classes.

|                | comp.windows.misc_windows.misc | comp.sys.mac.hardware | alt.atheism | rec.sport.baseball |
|----------------|---------------------------------|-----------------------|-------------|--------------------|
|                | rec.autos                       | alt.atheism           | rec.motorcycles | misc.forsale     |
| LBG            | 93.75%                          | 97.25%                | 94.125%     | 93%               |
| LDA            | 84.125%                         | 90.125%               | 58.25%      | 89%               |
| All            | 95.375%                         | 96.875%               | 88.875%     | 93%               |

Table 4: Unsupervised spam-filter classification performance comparison.

|       | LBG  | MOU  | pLSI |
|-------|------|------|------|
| Accuracy | 78%  | 60%  | 65%  |

Table 5: MovieLens Collaborative Filtering prediction results.

| Method          | Absolute Error |         |         |
|-----------------|----------------|---------|---------|
|                 | MAE            | RMSE    |         |
| Baseline        | 0.905          | 1.1445  |         |
| Gaussian pLSA   | 1.884          | 2.1142  |         |
| LBG             | 0.776          | 1.1183  |         |

We then moved to a supervised setting, in which we used the class labels (spam/ham) in the training stage. We modeled each class separately using LBG, LDA, MOU and pLSI while fitting the optimal number of topics per model (see note at the end of sec. 2). Note that MOU, when \( k = 1 \), reduces to the SpamBayes algorithm. The confusion table of each method is displayed in Fig. 2. Note the strikingly low false-positive (ham classified as spam) result for the LBG model, compared to other models. Future work might be directed to the development of an enhanced model, which will compensate for LBG’s limited success with false-negatives.

References

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 993–1022.

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering (pp. 43–52). Morgan Kaufmann.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society Ser. B*, 39.

Domingos, P., & Pazzani, M. J. (1997). On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29, 103–130.

Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI99* (pp. 289–296).

Hofmann, T. (2003). Collaborative filtering via gaussian probabilistic latent semantic analysis. *SIGIR* (pp. 259–266).

Lovasz, L., & Plummer, M. (1986). *Matching theory*. North Holland: Akadémiai Kiadó.

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. M. (2000). Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39, 103–134.