# 27 *Latent variable models for discrete data*

## 27.1 Introduction

In this chapter, we are concerned with latent variable models for discrete data, such as bit vectors, sequences of categorical variables, count vectors, graph structures, relational data, etc. These models can be used to analyze voting records, text and document collections, low-intensity images, movie ratings, etc. However, we will mostly focus on text analysis, and this will be reflected in our terminology.

Since we will be dealing with so many different kinds of data, we need some precise notation to keep things clear. When modeling variable-length sequences of categorical variables (i.e., symbols or **tokens**), such as words in a document, we will let $y_{il} \in \{1, \ldots, V\}$ represent the identity of the $l$'th word in document $i$, where $V$ is the number of possible words in the vocabulary. We assume $l = 1 : L_i$, where $L_i$ is the (known) length of document $i$, and $i = 1 : N$, where $N$ is the number of documents.

We will often ignore the word order, resulting in a **bag of words**. This can be reduced to a fixed length vector of counts (a histogram). We will use $n_{iv} \in \{0, 1, \ldots, L_i\}$ to denote the number of times word $v$ occurs in document $i$, for $v = 1 : V$. Note that the $N \times V$ count matrix $\mathbf{N}$ is often large but sparse, since we typically have many documents, but most words do not occur in any given document.

In some cases, we might have multiple different bags of words, e.g., bags of text words and bags of visual words. These correspond to different "channels" or types of features. We will denote these by $y_{irl}$, for $r = 1 : R$ (the number of responses) and $l = 1 : L_{ir}$. If $L_{ir} = 1$, it means we have a single token (a bag of length 1); in this case, we just write $y_{ir} \in \{1, \ldots, V_r\}$ for brevity. If every channel is just a single token, we write the fixed-size response vector as $\mathbf{y}_{i,1:R}$; in this case, the $N \times R$ design matrix $\mathbf{Y}$ will not be sparse. For example, in social science surveys, $y_{ir}$ could be the response of person $i$ to the $r$'th multi-choice question.

Out goal is to build joint probability models of $p(\mathbf{y}_i)$ or $p(\mathbf{n}_i)$ using latent variables to capture the correlations. We will then try to interpret the latent variables, which provide a compressed representation of the data. We provide an overview of some approaches in Section 27.2, before going into more detail in later sections.

Towards the end of the chapter, we will consider modeling graphs and relations, which can also be represented as sparse discrete matrices. For example, we might want to model the graph of which papers mycite which other papers. We will denote these relations by $\mathbf{R}$, reserving the symbol $\mathbf{Y}$ for any categorical data (e.g., text) associated with the nodes.

## 27.2 Distributed state LVMs for discrete data

In this section, we summarize a variety of possible approaches for constructing models of the form $p(\mathbf{y}_{i,1:L_i})$, for bags of tokens; $p(\mathbf{y}_{1:R})$, for vectors of tokens; and $p(\mathbf{n}_i)$, for vectors of integer counts.

### 27.2.1 Mixture models

The simplest approach is to use a finite mixture model (Chapter 11). This associates a single discrete latent variable, $q_i \in \{1, \ldots, K\}$, with every document, where $K$ is the number of clusters. We will use a discrete prior, $q_i \sim \text{Cat}(\boldsymbol{\pi})$. For variable length documents, we can define $p(y_{il}|q_i = k) = b_{kv}$, where $b_{kv}$ is the probability that cluster $k$ generates word $v$. The value of $q_i$ is known as a **topic**, and the vector $\mathbf{b}_k$ is the $k$'th topic's word distribution. That is, the likelihood has the form

$$p(\mathbf{y}_{i,1:L_i}|q_i = k) = \prod_{l=1}^{L_i} \text{Cat}(y_{il}|\mathbf{b}_k) \tag{27.1}$$

The induced distribution on the visible data is given by

$$p(\mathbf{y}_{i,1:L_i}) = \sum_k \pi_k \left[\prod_{l=1}^{L_i} \text{Cat}(y_{il}|\mathbf{b}_k)\right] \tag{27.2}$$

The "generative story" which this encodes is as follows: for document $i$, pick a topic $q_i$ from $\boldsymbol{\pi}$, call it $k$, and then for each word $l = 1 : L_i$, pick a word from $\mathbf{b}_k$. We will consider more sophisticated generative models later in this chapter.

If we have a fixed set of categorical observations, we can use a different topic matrix for each output variable:

$$p(\mathbf{y}_{i,1:R}|q_i = k) = \prod_{r=1}^{R} \text{Cat}(y_{il}|\mathbf{b}_k^{(r)}) \tag{27.3}$$

This is an unsupervised analog of naive Bayes classification.

We can also model count vectors. If the sum $L_i = \sum_v n_{iv}$ is known, we can use a multinomial:

$$p(\mathbf{n}_i|L_i, q_i = k) = \text{Mu}(\mathbf{n}_i|L_i, \mathbf{b}_k) \tag{27.4}$$

If the sum is unknown, we can use a Poisson class-conditional density to give

$$p(\mathbf{n}_i|q_i = k) = \prod_{v=1}^{V} \text{Poi}(n_{iv}|\lambda_{vk}) \tag{27.5}$$

In this case, $L_i|q_i = k \sim \text{Poi}(\sum_v \lambda_{vk})$.

### 27.2.2 Exponential family PCA

Unfortunately, finite mixture models are very limited in their expressive power. A more flexible model is to use a vector of real-valued continuous latent variables, similar to the factor analysis (FA) and PCA models in Chapter 12. In PCA, we use a Gaussian prior of the form $p(\mathbf{z}_i) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\mathbf{z}_i \in \mathbb{R}^K$, and a Gaussian likelihood of the form $p(\mathbf{y}_i|\mathbf{z}_i) = \mathcal{N}(\mathbf{W}\mathbf{z}_i, \sigma^2\mathbf{I})$. This method can certainly be applied to discrete or count data. Indeed, the method known as **latent semantic analysis** (**LSA**) or **latent semantic indexing** (**LSI**) (Deerwester et al. 1990; Dumais and Landauer 1997) is exactly equivalent to applying PCA to a term by document count matrix.

A better method for modeling categorical data is to use a multinoulli or multinomial distribution. We just have to change the likelihood to

$$p(\mathbf{y}_{i,1:L_i}|\mathbf{z}_i) = \prod_{l=1}^{L_i} \text{Cat}(y_{il}|\mathcal{S}(\mathbf{W}\mathbf{z}_i)) \tag{27.6}$$

where $\mathbf{W} \in \mathbb{R}^{V \times K}$ is a weight matrix and $\mathcal{S}$ is the softmax function. If we have a fixed number of categorical responses, we can use

$$p(\mathbf{y}_{1:R}|\mathbf{z}_i) = \prod_{r=1}^{R} \text{Cat}(y_{ir}|\mathcal{S}(\mathbf{W}_r\mathbf{z}_i)) \tag{27.7}$$

where $\mathbf{W}_r \in \mathbb{R}^{V \times K}$ is the weight matrix for the $r$'th response variable. This model is called **categorical PCA**, and is illustrated in Figure 27.1(a); see Section 12.4 for further discussion. If we have counts, we can use a multinomial model

$$p(\mathbf{n}_i|L_i, \mathbf{z}_i) = \text{Mu}(\mathbf{n}_i|L_i, \mathcal{S}(\mathbf{W}\mathbf{z}_i)) \tag{27.8}$$

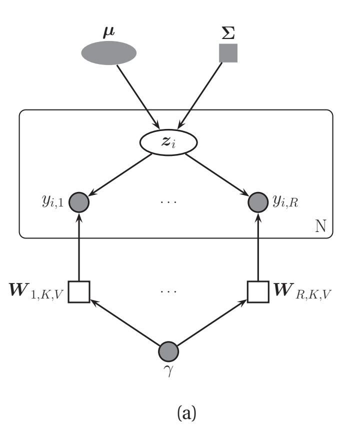
or a Poisson model

$$p(\mathbf{n}_i|\mathbf{z}_i) = \prod_{v=1}^{V} \text{Poi}(n_{iv}|\exp(\mathbf{w}_{v,:}^T\mathbf{z}_i)) \tag{27.9}$$

All of these models are examples of **exponential family PCA** or **ePCA** (Collins et al. 2002; Mohamed et al. 2008), which is an unsupervised analog of GLMs. The corresponding induced distribution on the visible variables has the form

$$p(\mathbf{y}_{i,1:L_i}) = \int \left[\prod_{l=1}^{L_i} p(y_{il}|\mathbf{z}_i, \mathbf{W})\right] \mathcal{N}(\mathbf{z}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{z}_i \tag{27.10}$$

Fitting this model is tricky, due to the lack of conjugacy. (Collins et al. 2002) proposed a coordinate ascent method that alternates between estimating the $\mathbf{z}_i$ and $\mathbf{W}$. This can be regarded as a degenerate version of EM, that computes a point estimate of $\mathbf{z}_i$ in the E step. The problem with the degenerate approach is that it is very prone to overfitting, since the number of latent variables is proportional to the number of datacases (Welling et al. 2008). A true EM algorithm would marginalize out the latent variables $\mathbf{z}_i$. A way to do this for categorical PCA, using variational EM, is discussed in Section 12.4. For more general models, one can use MCMC (Mohamed et al. 2008).



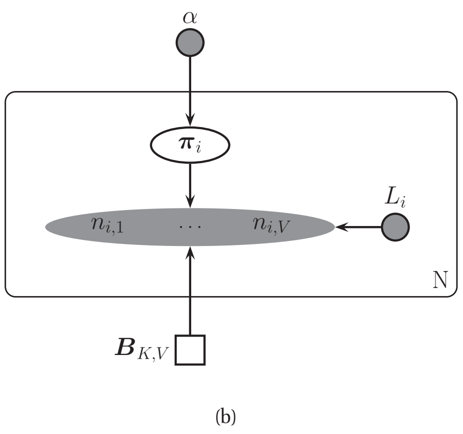


**Figure 27.1** Two LVMs for discrete data. Circles are scalar nodes, ellipses are vector nodes, squares are matrix nodes. (a) Categorical PCA. (b) Multinomial PCA.

### 27.2.3 LDA and mPCA

In ePCA, the quantity $\mathbf{W}\mathbf{z}_i$ represents the natural parameters of the exponential family. Sometimes it is more convenient to use the dual parameters. For example, for the multinomial, the dual parameter is the probability vector, whereas the natural parameter is the vector of log odds.

If we want to use the dual parameters, we need to constrain the latent variables so they live in the appropriate parameter space. In the case of categorical data, we will need to ensure the latent vector lives in $S_K$, the $K$-dimensional probability simplex. To avoid confusion with ePCA, we will denote such a latent vector by $\boldsymbol{\pi}_i$. In this case, the natural prior for the latent variables is the Dirichlet, $\boldsymbol{\pi}_i \sim \text{Dir}(\boldsymbol{\alpha})$. Typically we set $\boldsymbol{\alpha} = \alpha \mathbf{1}_K$. If we set $\alpha \ll 1$, we encourage $\boldsymbol{\pi}_i$ to be sparse, as shown in Figure 2.14.

When we have a count vector whose total sum is known, the likelihood is given by

$$p(\mathbf{n}_i|L_i, \boldsymbol{\pi}_i) = \text{Mu}(\mathbf{n}_i|L_i, \mathbf{B}\boldsymbol{\pi}_i) \tag{27.11}$$

This model is called **multinomial PCA** or **mPCA** (Buntine 2002; Buntine and Jakulin 2004, 2006). See Figure 27.1(b). Since we are assuming $n_{iv} = \sum_k b_{vk}\pi_{iv}$, this can be seen as a form of matrix factorization for the count matrix. Note that we use $b_{v,k}$ to denote the parameter vector, rather than $w_{v,k}$, since we impose the constraints that $0 \le b_{v,k} \le 1$ and $\sum_v b_{v,k} = 1$. The corresponding marginal distribution has the form

$$p(\mathbf{n}_i|L_i) = \int \text{Mu}(\mathbf{n}_i|L_i, \mathbf{B}\boldsymbol{\pi}_i)\text{Dir}(\boldsymbol{\pi}_i|\boldsymbol{\alpha})d\boldsymbol{\pi}_i \tag{27.12}$$

Unfortunately, this integral cannot be computed analytically.

If we have a variable length sequence (of known length), we can use

$$p(\mathbf{y}_{i,1:L_i}|\boldsymbol{\pi}_i) = \prod_{l=1}^{L_i} \text{Cat}(y_{il}|\mathbf{B}\boldsymbol{\pi}_i) \tag{27.13}$$

This is called **latent Dirichlet allocation** or **LDA** (Blei et al. 2003), and will be described in much greater detail below. LDA can be thought of as a probabilistic extension of LSA, where the latent quantities $\pi_{ik}$ are non-negative and sum to one. By contrast, in LSA, $z_{ik}$ can be negative which makes interpetation difficult.

A predecessor to LDA, known as **probabilistic latent semantic indexing** or **PLSI** (Hofmann 1999), uses the same model but computes a point estimate of $\boldsymbol{\pi}_i$ for each document (similar to ePCA), rather than integrating it out. Thus in PLSI, there is no prior for $\boldsymbol{\pi}_i$.

We can modify LDA to handle a fixed number of different categorical responses as follows:

$$p(\mathbf{y}_{i,1:R}|\boldsymbol{\pi}_i) = \prod_{r=1}^{R} \text{Cat}(y_{il}|\mathbf{B}^{(r)}\boldsymbol{\pi}_i) \tag{27.14}$$

This has been called the **user rating profile** (URP) model (Marlin 2003), and the **simplex factor model** (Bhattacharya and Dunson 2011).

### 27.2.4 GaP model and non-negative matrix factorization

Now consider modeling count vectors where we do not constrain the sum to be observed. In this case, the latent variables just need to be non-negative, so we will denote them by $\mathbf{z}_i^+$. This can be ensured by using a prior of the form

$$p(\mathbf{z}_i^+) = \prod_{k=1}^{K} \text{Ga}(z_{ik}^+|\alpha_k, \beta_k) \tag{27.15}$$

The likelihood is given by

$$p(\mathbf{n}_i|\mathbf{z}_i^+) = \prod_{v=1}^{V} \text{Poi}(n_{iv}|\mathbf{b}_{v,:}^T\mathbf{z}_i^+) \tag{27.16}$$

This is called the **GaP** (Gamma-Poisson) model (Canny 2004). See Figure 27.2(a).

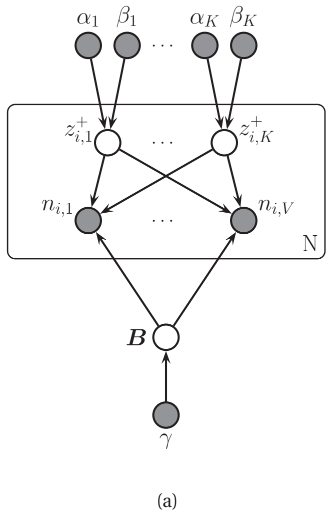
In (Buntine and Jakulin 2006), it is shown that the GaP model, when conditioned on a fixed $L_i$, reduces to the mPCA model. This follows since a set of Poisson random variables, when conditioned on their sum, becomes a multinomial distribution (see e.g., (Ross 1989)).

If we set $\alpha_k = \beta_k = 0$ in the GaP model, we recover a method known as **non-negative matrix factorization** or **NMF** (Lee and Seung 2001), as shown in (Buntine and Jakulin 2006). NMF is not a probabilistic generative model, since it does not specify a proper prior for $\mathbf{z}_i^+$. Furthermore, the algorithm proposed in (Lee and Seung 2001) is another degenerate EM algorithm, so suffers from overfitting. Some procedures to fit the GaP model, which overcome these problems, are given in (Buntine and Jakulin 2006).

To encourage $\mathbf{z}_i^+$ to be sparse, we can modify the prior to be a spike-and-Gamma type prior as follows:

$$p(z_{ik}^+) = \rho_k \mathbb{I}(z_{ik}^+ = 0) + (1 - \rho_k)\text{Ga}(z_{ik}^+|\alpha_k, \beta_k) \tag{27.17}$$

where $\rho_k$ is the probability of the spike at 0. This is called the conditional Gamma Poisson model (Buntine and Jakulin 2006). It is simple to modify Gibbs sampling to handle this kind of prior, although we will not go into detail here.



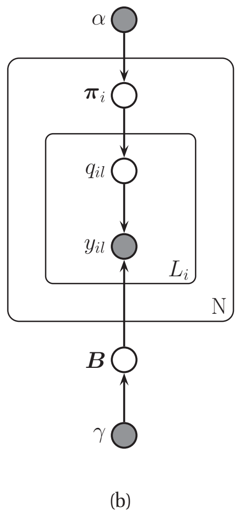


**Figure 27.2** (a) Gaussian-Poisson (GAP) model. (b) Latent Dirichlet allocation (LDA) model.

## 27.3 Latent Dirichlet allocation (LDA)

In this section, we explain the **latent Dirichlet allocation** or **LDA** (Blei et al. 2003) model in detail.

### 27.3.1 Basics

In a mixture of multinoullis, every document is assigned to a single topic, $q_i \in \{1, \ldots, K\}$, drawn from a global distribution $\boldsymbol{\pi}$. In LDA, every word is assigned to its own topic, $q_{il} \in \{1, \ldots, K\}$, drawn from a document-specific distribution $\boldsymbol{\pi}_i$. Since a document belongs to a distribution over topics, rather than a single topic, the model is called an **admixture mixture** or **mixed membership model** (Erosheva et al. 2004). This model has many other applications beyond text analysis, e.g., genetics (Pritchard et al. 2000), health science (Erosheva et al. 2007), social network analysis (Airoldi et al. 2008), etc.

Adding conjugate priors to the parameters, the full model is as follows:[1]

$$\boldsymbol{\pi}_i|\alpha \sim \mathrm{Dir}(\alpha \mathbf{1}_K) \tag{27.18}$$
$$q_{il}|\boldsymbol{\pi}_i \sim \mathrm{Cat}(\boldsymbol{\pi}_i) \tag{27.19}$$
$$\mathbf{b}_k|\gamma \sim \mathrm{Dir}(\gamma \mathbf{1}_V) \tag{27.20}$$
$$y_{il}|q_{il} = k, \mathbf{B} \sim \mathrm{Cat}(\mathbf{b}_k) \tag{27.21}$$

This is illustrated in Figure 27.2(b). We can marginalize out the $q_i$ variables, thereby creating a

1. Our notation is similar to the one we use elsewhere in this book, but is different from that used by most LDA papers. They typically use $w_{nd}$ for the identity of word $n$ in document $d$, $z_{nd}$ to represent the discrete indicator, $\boldsymbol{\theta}_d$ as the continuous latent vector for document $d$, and $\boldsymbol{\beta}_k$ as the $k$'th topic vector.

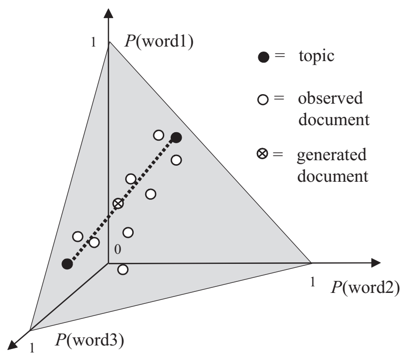


**Figure 27.3** Geometric interpretation of LDA. We have $K = 2$ topics and $V = 3$ words. Each document (white dots), and each topic (black dots), is a point in the 3d simplex. Source: Figure 5 of (Steyvers and Griffiths 2007). Used with kind permission of Tom Griffiths.

direct arc from $\boldsymbol{\pi}_i$ to $y_{il}$, with the following CPD:

$$p(y_{il} = v|\boldsymbol{\pi}_i) = \sum_k p(y_{il} = v|q_{il} = k)p(q_{il} = k) = \sum_k \pi_{ik} b_{kv} \tag{27.22}$$

As we mentioned in the introduction, this is very similar to the multinomial PCA model proposed in (Buntine 2002), which in turn is closely related to categorical PCA, GaP, NMF, etc.

LDA has an interesting geometric interpretation. Each vector $\mathbf{b}_k$ defines a distribution over $V$ words; each $k$ is known as a **topic**. Each document vector $\boldsymbol{\pi}_i$ defines a distribution over $K$ topics. So we model each document as an admixture over topics. Equivalently, we can think of LDA as a form of dimensionality reduction (assuming $K < V$, as is usually the case), where we project a point in the $V$-dimensional simplex (a normalized document count vector $\mathbf{x}_i$) onto the $K$-dimensional simplex. This is illustrated in Figure 27.3, where we have $V = 3$ words and $K = 2$ topics. The observed documents (which live in the 3d simplex) are approximated as living on a 2d simplex spanned by the 2 topic vectors, each of which lives in the 3d simplex.

One advantage of using the simplex as our latent space rather than Euclidean space is that the simplex can handle ambiguity. This is importance since in natural language, words can often have multiple meanings, a phenomomen known as **polysemy**. For example, “play” might refer to a verb (e.g., “to play ball” or “to play the coronet”), or to a noun (e.g., “Shakespeare’s play”). In LDA, we can have multiple topics, each of which can generate the word “play”, as shown in Figure 27.4, reflecting this ambiguity.

Given word $l$ in document $i$, we can compute $p(q_{il} = k|\mathbf{y}_i, \boldsymbol{\theta})$, and thus infer its most likely topic. By looking at the word in isolation, it might be hard to know what sense of the word is meant, but we can disambiguate this by looking at other words in the document. In particular, given $\mathbf{x}_i$, we can infer the topic distribution $\boldsymbol{\pi}_i$ for the document; this acts as a prior for disambiguating $q_{il}$. This is illustrated in Figure 27.5, where we show three documents from the TASA corpus.[2] In the first document, there are a variety of music related words, which suggest

2. The TASA corpus is a collection of 37,000 high-school level English documents, comprising over 10 million words,

Topic 77

| word | prob. |
|---|---|
| MUSIC | .090 |
| DANCE | .034 |
| SONG | .033 |
| **PLAY** | .030 |
| SING | .026 |
| SINGING | .026 |
| BAND | .026 |
| PLAYED | .023 |
| SANG | .022 |
| SONGS | .021 |
| DANCING | .020 |
| PIANO | .017 |
| PLAYING | .016 |
| RHYTHM | .015 |
| ALBERT | .013 |
| MUSICAL | .013 |

Topic 82

| word | prob. |
|---|---|
| LITERATURE | .031 |
| POEM | .028 |
| POETRY | .027 |
| POET | .020 |
| PLAYS | .019 |
| POEMS | .019 |
| **PLAY** | .015 |
| LITERARY | .013 |
| WRITERS | .013 |
| DRAMA | .012 |
| WROTE | .012 |
| POETS | .011 |
| WRITER | .011 |
| SHAKESPEARE | .010 |
| WRITTEN | .009 |
| STAGE | .009 |

Topic 166

| word | prob. |
|---|---|
| **PLAY** | .136 |
| BALL | .129 |
| GAME | .065 |
| PLAYING | .042 |
| HIT | .032 |
| PLAYED | .031 |
| BASEBALL | .027 |
| GAMES | .025 |
| BAT | .019 |
| RUN | .019 |
| THROW | .016 |
| BALLS | .015 |
| TENNIS | .011 |
| HOME | .010 |
| CATCH | .010 |
| FIELD | .010 |

**Figure 27.4** Three topics related to the word *play*. Source: Figure 9 of (Steyvers and Griffiths 2007). Used with kind permission of Tom Griffiths.

Document #29795

Bix beiderbecke, at age$^{060}$ fifteen$^{207}$, sat$^{174}$ on the slope$^{071}$ of a bluff$^{055}$ overlooking$^{027}$ the mississippi$^{137}$ river$^{137}$. He was listening$^{077}$ to music$^{077}$ coming$^{009}$ from a passing$^{043}$ riverboat. The music$^{077}$ had already captured$^{006}$ his heart$^{157}$ as well as his ear$^{119}$. It was jazz$^{077}$. Bix beiderbecke had already had music$^{077}$ lessons$^{077}$. He showed$^{002}$ promise$^{134}$ on the piano$^{077}$, and his parents$^{035}$ hoped$^{268}$ he might consider$^{118}$ becoming a concert$^{077}$ pianist$^{077}$. But bix was interested$^{268}$ in another kind$^{050}$ of music$^{077}$. He wanted$^{268}$ to $\boxed{\text{play}^{077}}$ the cornet. And he wanted$^{268}$ to $\boxed{\text{play}^{077}}$ jazz$^{077}$...

Document #1883

There is a simple$^{050}$ reason$^{106}$ why there are so few periods$^{078}$ of really great theater$^{082}$ in our whole western$^{046}$ world. Too many things$^{300}$ have to come right at the very same time. The dramatists must have the right actors$^{082}$, the actors$^{082}$ must have the right playhouses, the playhouses must have the right audiences$^{082}$. We must remember$^{288}$ that plays$^{082}$ exist$^{143}$ to be performed$^{077}$, not merely$^{050}$ to be read$^{254}$. ( even when you read$^{254}$ a $\boxed{\text{play}^{082}}$ to yourself, try$^{288}$ to perform$^{062}$ it, to put$^{174}$ it on a stage$^{078}$, as you go along.) as soon$^{028}$ as a $\boxed{\text{play}^{082}}$ has to be performed$^{082}$, then some kind$^{126}$ of theatrical$^{082}$...

Document #21359

Jim$^{296}$ has a game$^{166}$ book$^{254}$. Jim$^{296}$ reads$^{254}$ the book$^{254}$. Jim$^{296}$ sees$^{081}$ a game$^{166}$ for one. Jim$^{296}$ plays$^{166}$ the game$^{166}$. Jim$^{296}$ likes$^{081}$ the game$^{166}$ for one. The game$^{166}$ book$^{254}$ helps$^{081}$ jim$^{296}$. Don$^{180}$ comes$^{040}$ into the house$^{038}$. Don$^{180}$ and jim$^{296}$ read$^{254}$ the game$^{166}$ book$^{254}$. The boys$^{020}$ see a game$^{166}$ for two. The two boys$^{020}$ $\boxed{\text{play}^{166}}$ the game$^{166}$. The boys$^{020}$ $\boxed{\text{play}^{166}}$ the game$^{166}$ for two. The boys$^{020}$ like the game$^{166}$. Meg$^{282}$ comes$^{040}$ into the house$^{282}$. Meg$^{282}$ and don$^{180}$ and jim$^{296}$ read$^{254}$ the book$^{254}$. They see a game$^{166}$ for three. Meg$^{282}$ and don$^{180}$ and jim$^{296}$ $\boxed{\text{play}^{166}}$ the game$^{166}$. They $\boxed{\text{play}^{166}}$...

**Figure 27.5** Three documents from the TASA corpus containing different senses of the word *play*. Grayed out words were ignored by the model, because they correspond to uninteresting stop words (such as “and”, “the”, etc.) or very low frequency words. Source: Figure 10 of (Steyvers and Griffiths 2007). Used with kind permission of Tom Griffiths.

$\boldsymbol{\pi}_i$ will put most of its mass on the music topic (number 77); this in turn makes the music interpretation of "play" the most likely, as shown by the superscript. The second document interprets play in the theatrical sense, and the third in the sports sense. Note that is crucial that $\boldsymbol{\pi}_i$ be a latent variable, so information can flow between the $q_{il}$'s, thus enabling local disambiguation to use the full set of words.

## 27.3.2 Unsupervised discovery of topics

One of the main purposes of LDA is discover topics in a large collection or **corpus** of documents (see Figure 27.12 for an example). Unfortunately, since the model is unidentifiable, the interpertation of the topics can be difficult (Chang et al. 2009).. One approach, known as labeled LDA (Ramage et al. 2009), exploits the existence of tags on documents as a way to ensure identifiability. In particular, it forces the topics to correspond to the tags, and then it learns a distribution over words for each tag. This can make the results easier to interpret.

## 27.3.3 Quantitatively evaluating LDA as a language model

In order to evaluate LDA quantitatively, we can treat it as a **language model**, i.e., a probability distribution over sequences of words. Of course, it is not a very good language model, since it ignores word order and just looks at single words (unigrams), but it is interesting to compare LDA to other unigram-based models, such as mixtures of multinoullis, and pLSI. Such simple language models are sometimes useful for information retrieval purposes. The standard way to measure the quality of a language model is to use perplexity, which we now define below.

### 27.3.3.1 Perplexity

The **perplexity** of language model $q$ given a stochastic process[3] $p$ is defined as

$$\text{perplexity}(p, q) \triangleq 2^{H(p,q)} \tag{27.23}$$

where $H(p, q)$ is the cross-entropy of the two stochastic processes, defined as

$$H(p, q) \triangleq \lim_{N \to \infty} -\frac{1}{N} \sum_{\mathbf{y}_{1:N}} p(\mathbf{y}_{1:N}) \log q(\mathbf{y}_{1:N}) \tag{27.24}$$

The cross entropy (and hence perplexity) is minimized if $q = p$; in this case, the model can predict as well as the "true" distribution.

We can approximate the stochastic process by using a single long test sequence (composed of multiple documents and multiple sentences, complete with end-of-sentence markers), call it $\mathbf{y}^*_{1:N}$. (This approximation becomes more and more accurate as the sequence gets longer, provided the process is stationary and ergodic (Cover and Thomas 2006).) Define the empirical distribution (an approximation to the stochastic process) as

$$p_{\text{emp}}(\mathbf{y}_{1:N}) = \delta_{\mathbf{y}^*_{1:N}}(\mathbf{y}_{1:N}) \tag{27.25}$$

collated by a company formerly known as Touchstone Applied Science Associates, but now known as Questar Assessment Inc `www.questarai.com`.

3. A stochastic process is one which can define a joint distribution over an arbitrary number of random variables. We can think of natural language as a stochastic process, since it can generate an infinite stream of words.

In this case, the cross-entropy becomes

$$H(p_{\text{emp}}, q) = -\frac{1}{N}\log q(\mathbf{y}^*_{1:N}) \tag{27.26}$$

and the perplexity becomes

$$\text{perplexity}(p_{\text{emp}}, q) = 2^{H(p_{\text{emp}}, q)} = q(\mathbf{y}^*_{1:N})^{-1/N} = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{q(y_i^*|\mathbf{y}^*_{1:i-1})}} \tag{27.27}$$

We see that this is the geometric mean of the inverse predictive probabilities, which is the usual definition of perplexity (Jurafsky and Martin 2008, p96).

In the case of unigram models, the cross entropy term is given by

$$H = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{L_i}\sum_{l=1}^{L_i}\log q(y_{il}) \tag{27.28}$$

where $N$ is the number of documents and $L_i$ is the number of words in document $i$. Hence the perplexity of model $q$ is given by

$$\text{perplexity}(p_{\text{emp}}, p) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\frac{1}{L_i}\sum_{l=1}^{L_i}\log q(y_{il})\right) \tag{27.29}$$

Intuitively, perplexity mesures the weighted average **branching factor** of the model's predictive distribution. Suppose the model predicts that each symbol (letter, word, whatever) is equally likely, so $p(y_i|\mathbf{y}_{1:i-1}) = 1/K$. Then the perplexity is $((1/K)^N)^{-1/N} = K$. If some symbols are more likely than others, and the model correctly reflects this, its perplexity will be lower than $K$. Of course, $H(p, p) = H(p) \leq H(p, q)$, so we can never reduce the perplexity below the entropy of the underlying stochastic process.

#### 27.3.3.2 Perplexity of LDA

The key quantity is $p(v)$, the predictive distribution of the model over possible words. (It is implicitly conditioned on the training set.) For LDA, this can be approximated by plugging in $\mathbf{B}$ (e.g., the posterior mean estimate) and approximately integrating out $\mathbf{q}$ using mean field inference (see (Wallach et al. 2009) for a more accurate way to approximate the predictive likelihood).

In Figure 27.6, we compare LDA to several other simple unigram models, namely MAP estimation of a multinoulli, MAP estimation of a mixture of multinoullis, and pLSI. (When performing MAP estimation, the same Dirichlet prior on $\mathbf{B}$ was used as in the LDA model.) The metric is perplexity, as in Equation 27.29, and the data is a subset of the TREC AP corpus containing 16,333 newswire articles with 23,075 unique terms. We see that LDA significantly outperforms these other methods.

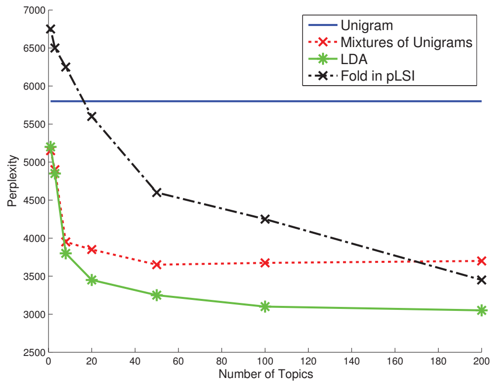


**Figure 27.6** Perplexity vs number of topics on the TREC AP corpus for various language models. Based on Figure 9 of (Blei et al. 2003). Figure generated by `bleiLDAperplexityPlot`.

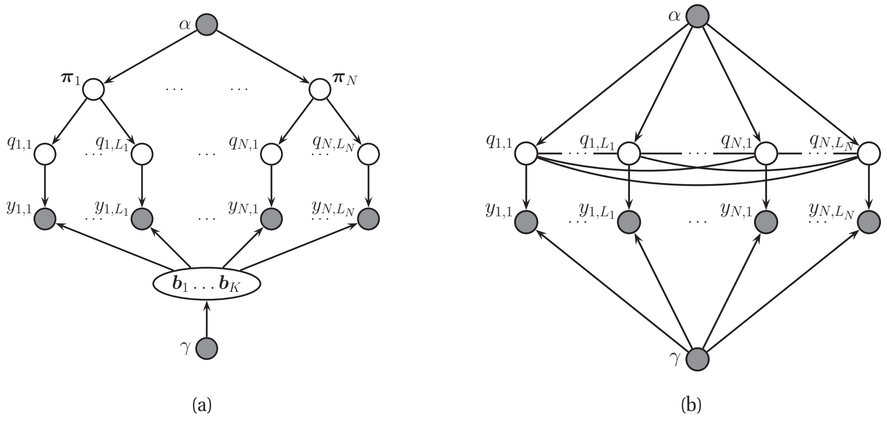


**Figure 27.7** (a) LDA unrolled for $N$ documents. (b) Collapsed LDA, where we integrate out the $\boldsymbol{\pi}_i$ and the $\mathbf{b}_k$.

### 27.3.4 Fitting using (collapsed) Gibbs sampling

It is straightforward to derive a Gibbs sampling algorithm for LDA. The full conditionals are as follows:

$$p(q_{il} = k|\cdot) \propto \exp[\log \pi_{ik} + \log b_{k,x_{il}}] \tag{27.30}$$

$$p(\boldsymbol{\pi}_i|\cdot) = \text{Dir}(\{\alpha_k + \sum_l \mathbb{I}(z_{il} = k)\}) \tag{27.31}$$

$$p(\mathbf{b}_k|\cdot) = \text{Dir}(\{\gamma_v + \sum_i \sum_l \mathbb{I}(x_{il} = v, z_{il} = k)\}) \tag{27.32}$$

However, one can get better performance by analytically integrating out the $\boldsymbol{\pi}_i$'s and the $\mathbf{b}_k$'s, both of which have a Dirichlet distribution, and just sampling the discrete $q_{il}$'s. This approach was first suggested in (Griffiths and Steyvers 2004), and is an example of **collapsed Gibbs sampling**. Figure 27.7(b) shows that now all the $q_{il}$ variables are fully correlated. However, we can sample them one at a time, as we explain below.

First, we need some notation. Let $c_{ivk} = \sum_{l=1}^{L_i} \mathbb{I}(q_{il} = k, y_{il} = v)$ be the number of times word $v$ is assigned to topic $k$ in document $i$. Let $c_{ik} = \sum_v c_{ivk}$ be the number of times any word from document $i$ has been assigned to topic $k$. Let $c_{vk} = \sum_i c_{ivk}$ be the number of times word $v$ has been assigned to topic $k$ in any document. Let $n_{iv} = \sum_k c_{ivk}$ be the number of times word $v$ occurs in document $i$; this is observed. Let $c_k = \sum_v c_{vk}$ be the number of words assigned to topic $k$. Finally, let $L_i = \sum_k c_{ik}$ be the number of words in document $i$; this is observed.

We can now derive the marginal prior. By applying Equation 5.24, one can show that

$$p(\mathbf{q}|\alpha) = \prod_i \int \left[ \prod_{l=1}^{L_i} \mathrm{Cat}(q_{il}|\boldsymbol{\pi}_i) \right] \mathrm{Dir}(\boldsymbol{\pi}_i|\boldsymbol{\alpha}\mathbf{1}_K) d\boldsymbol{\pi}_i \tag{27.33}$$

$$= \left( \frac{\Gamma(K\alpha)}{\Gamma(\alpha)^K} \right)^N \prod_{i=1}^N \frac{\prod_{k=1}^K \Gamma(c_{ik} + \alpha)}{\Gamma(L_i + K\alpha)} \tag{27.34}$$

By similar reasoning, one can show

$$p(\mathbf{y}|\mathbf{q}, \gamma) = \prod_k \int \left[ \prod_{il:q_{il}=k} \mathrm{Cat}(y_{il}|\mathbf{b}_k) \right] \mathrm{Dir}(\mathbf{b}_k|\gamma \mathbf{1}_V) d\mathbf{b}_k \tag{27.35}$$

$$= \left( \frac{\Gamma(V\beta)}{\Gamma(\beta)^V} \right)^K \prod_{k=1}^K \frac{\prod_{v=1}^V \Gamma(c_{vk} + \beta)}{\Gamma(c_k + V\beta)} \tag{27.36}$$

From the above equations, and using the fact that $\Gamma(x+1)/\Gamma(x) = x$, we can derive the full conditional for $p(q_{il}|\mathbf{q}_{-i,l})$. Define $c_{ivk}^-$ to be the same as $c_{ivk}$ except it is compute by summing over all locations in document $i$ except for $q_{il}$. Also, let $y_{il} = v$. Then

$$p(q_{i,l} = k|\mathbf{q}_{-i,l}, \mathbf{y}, \alpha, \gamma) \propto \frac{c_{v,k}^- + \gamma}{c_k^- + V\gamma} \frac{c_{i,k}^- + \alpha}{L_i + K\alpha} \tag{27.37}$$

We see that a word in a document is assigned to a topic based both on how often that word is generated by the topic (first term), and also on how often that topic is used in that document (second term).

Given Equation 27.37, we can implement the collapsed Gibbs sampler as follows. We randomly assign a topic to each word, $q_{il} \in \{1, \ldots, K\}$. We can then sample a new topic as follows: for a given word in the corpus, decrement the relevant counts, based on the topic assigned to the current word; draw a new topic from Equation 27.37, update the count matrices; and repeat. This algorithm can be made efficient since the count matrices are very sparse.

### 27.3.5 Example

This process is illustrated in Figure 27.8 on a small example with two topics, and five words. The left part of the figure illustrates 16 documents that were sampled from the LDA model using

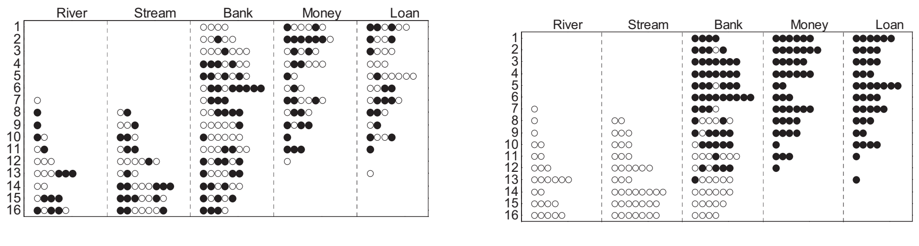


**Figure 27.8** Illustration of (collapsed) Gibbs sampling applied to a small LDA example. There are $N = 16$ documents, each containing a variable number of words drawn from a vocabulary of $V = 5$ words, There are two topics. A white dot means word the word is assigned to topic 1, a black dot means the word is assigned to topic 2. (a) The initial random assignment of states. (b) A sample from the posterior after 64 steps of Gibbs sampling. Source: Figure 7 of (Steyvers and Griffiths 2007). Used with kind permission of Tom Griffiths.

$p(\text{money}|k = 1) = p(\text{loan}|k = 1) = p(\text{bank}|k = 1) = 1/3$ and $p(\text{river}|k = 2) = p(\text{stream}|k = 2) = p(\text{bank}|k = 2) = 1/3$. For example, we see that the first document contains the word "bank" 4 times (indicated by the four dots in row 1 of the "bank" column), as well as various other financial terms. The right part of the figure shows the state of the Gibbs sampler after 64 iterations. The "correct" topic has been assigned to each token in most cases. For example, in document 1, we see that the word "bank" has been correctly assigned to the financial topic, based on the presence of the words "money" and "loan". The posterior mean estimate of the parameters is given by $\hat{p}(\text{money}|k = 1) = 0.32$, $\hat{p}(\text{loan}|k = 1) = 0.29$, $\hat{p}(\text{bank}|k = 1) = 0.39$, $\hat{p}(\text{river}|k = 2) = 0.25$, $\hat{p}(\text{stream}|k = 2) = 0.4$, and $\hat{p}(\text{bank}|k = 2) = 0.35$, which is impressively accurate, given that there are only 16 training examples.

### 27.3.6 Fitting using batch variational inference

A faster alternative to MCMC is to use variational EM. (We cannot use exact EM since exact inference of $\boldsymbol{\pi}_i$ and $\mathbf{q}_i$ is intractable.) We give the details below.

#### 27.3.6.1 Sequence version

Following (Blei et al. 2003), we will use a fully factorized (mean field) approximation of the form

$$q(\boldsymbol{\pi}_i, \mathbf{q}_i) = \text{Dir}(\boldsymbol{\pi}_i|\tilde{\boldsymbol{\pi}}_i) \prod_l \text{Cat}(q_{il}|\tilde{\mathbf{q}}_{il}) \tag{27.38}$$

We will follow the usual mean field recipe. For $q(q_{il})$, we use Bayes' rule, but where we need to take expectations over the prior:

$$\tilde{q}_{ilk} \quad \propto \quad b_{y_{i,l},k} \exp(\mathbb{E}[\log \pi_{ik}]) \tag{27.39}$$

where

$$\mathbb{E}[\log \pi_{ik}] = \psi_k(\tilde{\boldsymbol{\pi}}_{i\cdot}) \triangleq \Psi(\tilde{\pi}_{ik}) - \Psi(\sum_{k'} \tilde{\pi}_{ik'}) \tag{27.40}$$

where $\Psi$ is the digamma function. The update for $q(\boldsymbol{\pi}_i)$ is obtained by adding up the expected counts:

$$\tilde{\pi}_{ik} = \alpha_k + \sum_l \tilde{q}_{ilk} \tag{27.41}$$

The M step is obtained by adding up the expected counts and normalizing:

$$\hat{b}_{vk} \propto \gamma_v + \sum_{i=1}^{N} \sum_{l=1}^{L_i} \tilde{q}_{ilk} \mathbb{I}(y_{il} = v) \tag{27.42}$$

#### 27.3.6.2 Count version

Note that the E step takes $O((\sum_i L_i)VK)$ space to store the $\tilde{q}_{ilk}$. It is much more space efficient to perform inference in the mPCA version of the model, which works with counts; these only take $O(NVK)$ space, which is a big savings if documents are long. (By contrast, the collapsed Gibbs sampler must work explicitly with the $q_{il}$ variables.)

We will focus on approximating $p(\boldsymbol{\pi}_i, \mathbf{c}_i | \mathbf{n}_i, L_i)$, where we write $\mathbf{c}_i$ as shorthand for $\mathbf{c}_{i..}$. We will again use a fully factorized (mean field) approximation of the form

$$q(\boldsymbol{\pi}_i, \mathbf{c}_i) = \text{Dir}(\boldsymbol{\pi}_i | \tilde{\boldsymbol{\pi}}_i) \prod_v \text{Mu}(\mathbf{c}_{iv.} | n_{iv}, \tilde{\mathbf{c}}_{iv.}) \tag{27.43}$$

The new E step becomes

$$\tilde{\pi}_{ik} = \alpha_k + \sum_v n_{iv} \tilde{c}_{ivk} \tag{27.44}$$

$$\tilde{c}_{ivk} \propto b_{vk} \exp(\mathbb{E}[\log \pi_{ik}]) \tag{27.45}$$

The new M step becomes

$$\hat{b}_{vk} \propto \gamma_v + \sum_i n_{iv} \tilde{c}_{ivk} \tag{27.46}$$

#### 27.3.6.3 VB version

We now modify the algorithm to use VB instead of EM, so that we infer the parameters as well as the latent variables. There are two advantages to this. First, by setting $\gamma \ll 1$, VB will encourage $\mathbf{B}$ to be sparse (as in Section 21.6.1.6). Second, we will be able to generalize this to the online learning setting, as we discuss below.

Our new posterior approximation becomes

$$q(\boldsymbol{\pi}_i, \mathbf{c}_i, \mathbf{B}) = \text{Dir}(\boldsymbol{\pi}_i | \tilde{\boldsymbol{\pi}}_i) \prod_v \text{Mu}(\mathbf{c}_{iv.} | n_{iv}, \tilde{\mathbf{c}}_{iv.}) \prod_k \text{Dir}(\mathbf{b}_{.k} | \tilde{\mathbf{b}}_{.k}) \tag{27.47}$$

The update for $\tilde{c}_{ivk}$ changes, to the following:

$$\tilde{c}_{ivk} \propto \exp\left(\mathbb{E}[\log b_{vk}] + \mathbb{E}[\log \pi_{ik}]\right) \tag{27.48}$$

**Algorithm 27.1:** Batch VB for LDA

```
1  Input: $n_{iv}$, $K$, $\alpha_k$, $\gamma_v$;
2  Estimate $\tilde{b}_{vk}$ using EM for multinomial mixtures;
3  Initialize counts $n_{iv}$;
4  while not converged do
5      // E step ;
6      $s_{vk} = 0$ // expected sufficient statistics;
7      for each document $i = 1 : N$ do
8          $(\tilde{\boldsymbol{\pi}}_i, \tilde{\mathbf{c}}_i) = \text{Estep}(\mathbf{n}_i, \tilde{\mathbf{B}}, \boldsymbol{\alpha})$;
9          $s_{vk} += n_{iv}\tilde{c}_{ivk}$;
10     // M step ;
11     for each topic $k = 1 : K$ do
12         $\tilde{b}_{vk} = \gamma_v + s_{vk}$;
13 function $(\tilde{\boldsymbol{\pi}}_i, \tilde{\mathbf{c}}_i) = \text{Estep}(\mathbf{n}_i, \tilde{\mathbf{B}}, \boldsymbol{\alpha})$;
14 Initialize $\tilde{\pi}_{ik} = \alpha_k$;
15 repeat
16     $\tilde{\pi}_{i.}^{old} = \tilde{\pi}_{i.}$, $\tilde{\pi}_{ik} = \alpha_k$;
17     for each word $v = 1 : V$ do
18         for each topic $k = 1 : K$ do
19             $\tilde{c}_{ivk} = \exp\left(\psi_k(\tilde{\mathbf{b}}_{v.}) + \psi_k(\tilde{\boldsymbol{\pi}}_{i.}^{old})\right)$;
20         $\tilde{\mathbf{c}}_{iv.} = \text{normalize}(\tilde{\mathbf{c}}_{iv.})$;
21         $\tilde{\pi}_{ik} += n_{iv}\tilde{c}_{ivk}$
22 until $\frac{1}{K}\sum_k |\tilde{\pi}_{ik} - \tilde{\pi}_{ik}^{old}| <$ thresh;
```

Also, the M step becomes

$$\tilde{b}_{vk} = \gamma_v + \sum_i \tilde{c}_{ivk} \tag{27.49}$$

No normalization is required, since we are just updating the pseudcounts. The overall algorithm is summarized in Algorithm 22.

### 27.3.7 Fitting using online variational inference

In the bathc version, the E step clearly takes $O(NKVT)$ time, where $T$ is the number of mean field updates (typically $T \sim 5$). This can be slow if we have many documents. This can be reduced by using stochastic gradient descent (Section 8.5.2) to perform online variational inference, as we now explain.

We can derive an online version, following (Hoffman et al. 2010). We perform an E step in the usual way. We then compute the variational parameters for $\mathbf{B}$ treating the expected sufficient statistics from the single data case as if the whole data set had those statistics. Finally, we make

**Algorithm 27.2:** Online variational Bayes for LDA

```
1 Input: $n_{iv}$, $K$, $\alpha_k$, $\gamma_v$, $\tau_0$, $\kappa$;
2 Initialize $\tilde{b}_{vk}$ randomly;
3 for $t = 1 : \infty$ do
4     Set step size $\rho_t = (\tau_0 + t)^{-\kappa}$;
5     Pick document $i = i(t)$; ;
6     $(\tilde{\boldsymbol{\pi}}_i, \tilde{\mathbf{c}}_i) = \text{Estep}(\mathbf{n}_i, \tilde{\mathbf{B}}, \boldsymbol{\alpha})$;
7     $\tilde{b}_{vk}^{new} = \gamma_v + N n_{iv} \tilde{c}_{ivk}$;
8     $\tilde{b}_{vk} = (1 - \rho_t)\tilde{b}_{vk} + \rho_t \tilde{b}_{vk}^{new}$;
```

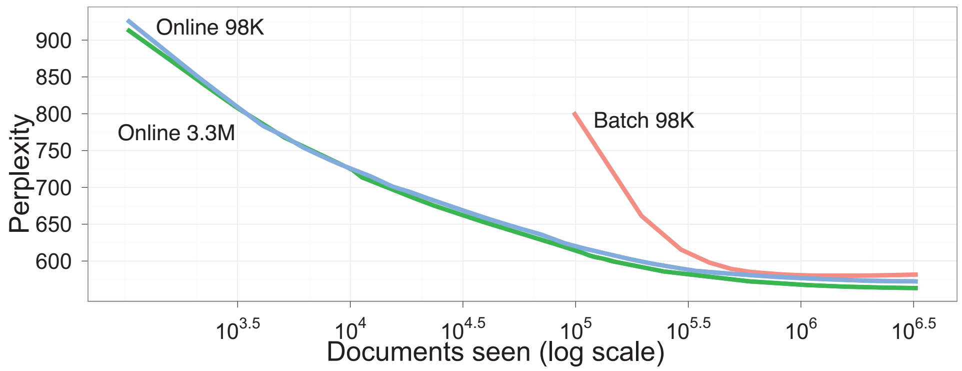


**Figure 27.9** Test perplexity vs number of training documents for batch and online VB-LDA. From Figure 1 of (Hoffman et al. 2010). Used with kind permission of David Blei.

a partial update for the variational parameters for $\mathbf{B}$, putting weight $\rho_t$ on the new estimate and weight $1 - \rho_t$ on the old estimate. The step size $\rho_t$ decays over time, as in Equation 8.83. The overall algorithm is summarized in Algorithm 3. In practice, we should use mini-batches, as explained in Section 8.5.2.3. In (Hoffman et al. 2010), they used a batch of size 256–4096.

Figure 27.9 plots the perplexity on a test set of size 1000 vs number of analyzed documents (E steps), where the data is drawn from (English) Wikipedia. The figure shows that online variational inference is much faster than offline inference, yet produces similar results.

## 27.3.8 Determining the number of topics

Choosing $K$, the number of topics, is a standard model selection problem. Here are some approaches that have been taken:

- Use annealed importance sampling (Section 24.6.2) to approximate the evidence (Wallach et al. 2009).
- Cross validation, using the log likelihood on a test set.

- Use the variational lower bound as a proxy for $\log p(\mathcal{D}|K)$.
- Use non-parametric Bayesian methods (Teh et al. 2006).

## 27.4 Extensions of LDA

Many extensions of LDA have been proposed since the first paper came out in 2003. We briefly discuss a few of these below.

### 27.4.1 Correlated topic model

One weakness of LDA is that it cannot capture correlation between topics. For example, if a document has the "business" topic, it is reasonable to expect the "finance" topic to co-occcur. The source of the problem is the use of a Dirichlet prior for $\boldsymbol{\pi}_i$. The problem with the Dirichelt it that it is characterized by just a mean vector and a strength parameter, but its covariance is fixed ($\Sigma_{ij} = -\alpha_i \alpha_j$), rather than being a free parameter.

One way around this is to replace the Dirichlet prior with the logistic normal distribution, as in categorical PCA (Section 27.2.2). The model becomes

$$\begin{aligned}
\mathbf{b}_k|\gamma &\sim \mathrm{Dir}(\gamma \mathbf{1}_V) && (27.50)\\
\mathbf{z}_i &\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) && (27.51)\\
\boldsymbol{\pi}_i|\mathbf{z}_i &= \mathcal{S}(\mathbf{z}_i) && (27.52)\\
q_{il}|\boldsymbol{\pi}_i &\sim \mathrm{Cat}(\boldsymbol{\pi}_i) && (27.53)\\
y_{il}|q_{il} = k, \mathbf{B} &\sim \mathrm{Cat}(\mathbf{b}_k) && (27.54)
\end{aligned}$$

This is known as the **correlated topic model** (Blei and Lafferty 2007). This is very similar to categorical PCA, but slightly different. To see the difference, let us marginalize out the $q_{il}$ and $\boldsymbol{\pi}_i$. Then in the CTM we have

$$y_{il} \sim \mathrm{Cat}(\mathbf{B}\mathcal{S}(\mathbf{z}_i)) \tag{27.55}$$

where $\mathbf{B}$ is a stochastic matrix. By contrast, in catPCA we have

$$y_{il} \sim \mathrm{Cat}(\mathcal{S}(\mathbf{W}\mathbf{z}_i)) \tag{27.56}$$

where $\mathbf{W}$ is an unconstrained matrix.

Fitting this model is tricky, since the prior for $\boldsymbol{\pi}_i$ is no longer conjugate to the multinomial likelihood for $q_{il}$. However, we can use any of the variational methods in Section 21.8.1.1, where we discussed Bayesian multiclass logistic regression. In the CTM case, things are even harder since the categorical response variables $\mathbf{q}_i$ are hidden, but we can handle this by using an additional mean field approximation. See (Blei and Lafferty 2007) for details.

Having fit the model, one can then convert $\hat{\boldsymbol{\Sigma}}$ to a sparse precision matrix $\hat{\boldsymbol{\Sigma}}^{-1}$ by pruning low-strength edges, to get a sparse Gaussian graphical model. This allows you to visualize the correlation between topics. Figure 27.10 shows the result of applying this procedure to articles from *Science* magazine, from 1990-1999. (This corpus contains 16,351 documents, and 5.7M words (19,088 of them unique), after stop-word and low-frequency removal.) Nodes represent topics, with the top 5 words per topic listed inside. The font size reflects the overall prevalence of the topic in the corpus. Edges represent significant elements of the precision matrix.

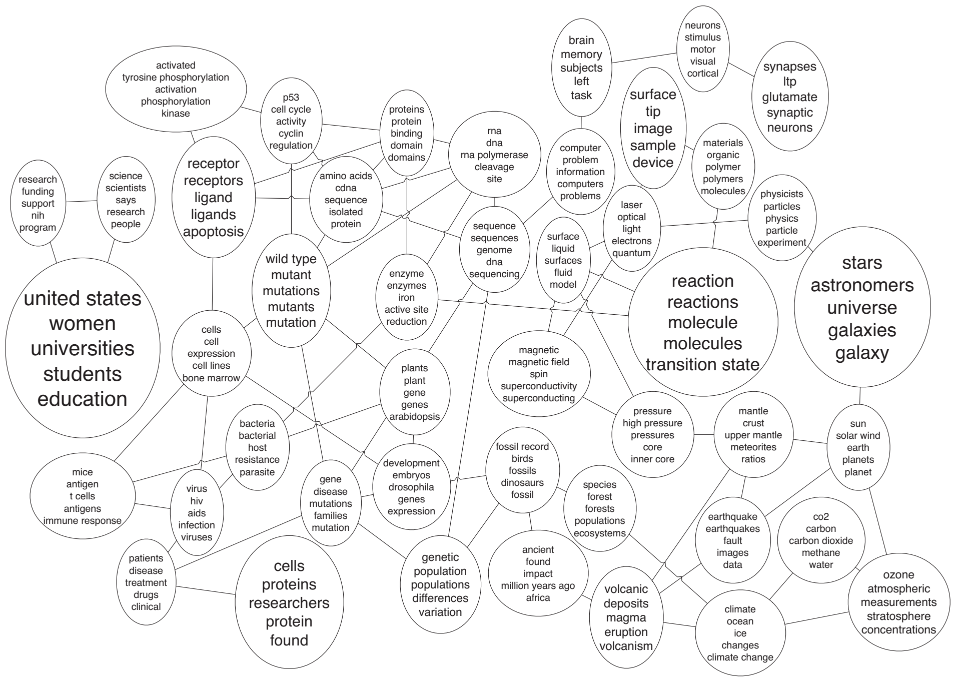

activated tyrosine phosphorylation activation phosphorylation kinase
p53 cell cycle activity cyclin regulation
proteins protein binding domain domains
rna dna rna polymerase cleavage site
brain memory subjects left task
neurons stimulus motor visual cortical
synapses ltp glutamate synaptic neurons
surface tip image sample device
computer problem information computers problems
materials organic polymer polymers molecules
research funding support nih program
science scientists says research people
receptor receptors ligand ligands apoptosis
amino acids cdna sequence isolated protein
sequence sequences genome dna sequencing
surface liquid surfaces fluid model
laser optical light electrons quantum
physicists particles physics particle experiment
united states women universities students education
wild type mutant mutations mutants mutation
enzyme enzymes iron active site reduction
reaction reactions molecule molecules transition state
stars astronomers universe galaxies galaxy
cells cell expression cell lines bone marrow
magnetic magnetic field spin superconductivity superconducting
plants plant gene genes arabidopsis
bacteria bacterial host resistance parasite
pressure high pressure pressures core inner core
mantle crust upper mantle meteorites ratios
sun solar wind earth planets planet
mice antigen t cells antigens immune response
virus hiv aids infection viruses
gene disease mutations families mutation
development embryos drosophila genes expression
fossil record birds fossils dinosaurs fossil
species forest forests populations ecosystems
earthquake earthquakes fault images data
co2 carbon carbon dioxide methane water
ozone atmospheric measurements stratosphere concentrations
patients disease treatment drugs clinical
cells proteins researchers protein found
genetic population populations differences variation
ancient found impact million years ago africa
volcanic deposits magma eruption volcanism
climate ocean ice changes climate change

**Figure 27.10** Output of the correlated topic model (with $K = 50$ topics) when applied to articles from *Science*. Nodes represent topics, with the 5 most probable phrases from each topic shown inside. Font size reflects overall prevalence of the topic. See `http://www.cs.cmu.edu/~lemur/science/` for an interactive version of this model with 100 topics. Source: Figure 2 of (Blei and Lafferty 2007). Used with kind permission of David Blei.

## 27.4.2 Dynamic topic model

In LDA, the topics (distributions over words) are assumed to be static. In some cases, it makes sense to allow these distributions to evolve smoothly over time. For example, an article might use the topic "neuroscience", but if it was written in the 1900s, it is more likely to use words like "nerve", whereas if it was written in the 2000s, it is more likely to use words like "calcium receptor" (this reflects the general trend of neuroscience towards molecular biology).

One way to model this is use a dynamic logistic normal model, as illustrated in Figure 27.11. In particular, we assume the topic distributions evolve according to a Gaussian random walk, and then we map these Gaussian vectors to probabilities via the softmax function:

$$\mathbf{b}_{t,k}|\mathbf{b}_{t-1,k} \sim \mathcal{N}(\mathbf{b}_{t-1,k}, \sigma^2 \mathbf{1}_V) \tag{27.57}$$

$$\boldsymbol{\pi}_i^t \sim \text{Dir}(\alpha \mathbf{1}_K) \tag{27.58}$$

$$q_{il}^t|\boldsymbol{\pi}_i^t \sim \text{Cat}(\boldsymbol{\pi}_i^t) \tag{27.59}$$

$$y_{il}^t|q_{il}^t = k, \mathbf{B}^t \sim \text{Cat}(\mathcal{S}(\mathbf{b}_k^t)) \tag{27.60}$$

This is known as a **dynamic topic model** (Blei and Lafferty 2006b).

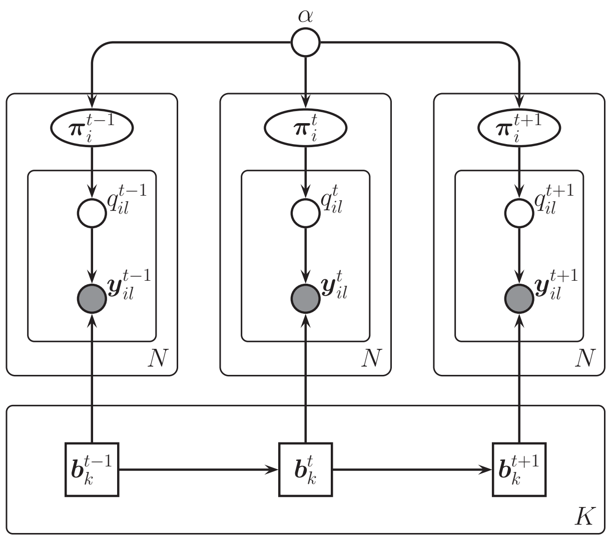


**Figure 27.11** The dynamic topic model.

One can perform approximate infernece in this model using a structured mean field method (Section 21.4), that exploits the Kalman smoothing algorithm (Section 18.3.1) to perform exact inference on the linear-Gaussian chain between the $\mathbf{b}_{t,k}$ nodes (see (Blei and Lafferty 2006b) for details).

Figure 27.12 illustrates a typical output of the system when applied to 100 years of articles from *Science*. On the top, we visualize the top 10 words from a specific topic (which seems to be related to neuroscience) after 10 year intervals. On the bottom left, we plot the probability of some specific words belonging to this topic. On the bottom right, we list the titles of some articles that contained this topic.

One interesting application of this model is to perform temporally-corrected document retrieval. That is, suppose we look for documents about the inheritance of disease. Modern articles will use words like "DNA", but older articles (before the discovery of DNA) may use other terms such as "heritable unit". But both articles are likely to use the same topics. Similar ideas can be used to perform cross-language information retrieval, see e.g., (Cimiano et al. 2009).

### 27.4.3 LDA-HMM

The LDA model assumes words are exchangeable, which is clearly not true. A simple way to model sequential dependence between words is to use a hidden Markov model or HMM. The trouble with HMMs is that they can only model short-range dependencies, so they cannot capture the overall gist of a document. Hence they can generate syntactically correct sentences (see e.g., Table 17.1). but not semantically plausible ones.

It is possible to combine LDA with HMM to create a model called **LDA-HMM** (Griffiths et al.

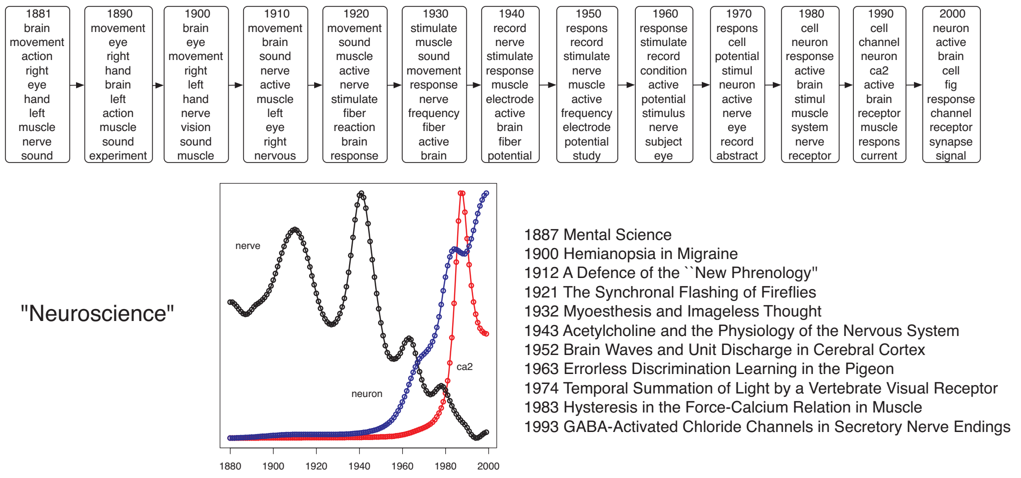


**Figure 27.12** Part of the output of the dynamic topic model when applied to articles from *Science*. We show the top 10 words for the neuroscience topic over time. We also show the probability of three words within this topic over time, and some articles that contained this topic. Source: Figure 4 of (Blei and Lafferty 2006b). Used with kind permission of David Blei.

2004). This model uses the HMM states to model function or syntactic words, such as "and" or "however", and uses the LDA to model content or semantic words, which are harder to predict. There is a distinguished HMM state which specifies when the LDA model should be used to generate the word; the rest of the time, the HMM generates the word.

More formally, for each document $i$, the model defines an HMM with states $z_{il} \in \{0, \ldots, C\}$. In addition, each document has an LDA model associated with it. If $z_{il} = 0$, we generate word $y_{il}$ from the semantic LDA model, with topic specified by $q_{il}$; otherwise we generate word $y_{il}$ from the syntactic HMM model. The DGM is shown in Figure 27.13. The CPDs are as follows:

$$p(\boldsymbol{\pi}_i) = \text{Dir}(\boldsymbol{\pi}_i|\alpha\mathbf{1}_K) \tag{27.61}$$

$$p(q_{il} = k|\boldsymbol{\pi}_i) = \pi_{ik} \tag{27.62}$$

$$p(z_{il} = c'|z_{i,l-1} = c) = A^{HMM}(c, c') \tag{27.63}$$

$$p(y_{il} = v|q_{il} = k, z_{il} = c) = \begin{cases} B^{LDA}(k, v) & \text{if } c = 0 \\ B^{HMM}(c, v) & \text{if } c > 0 \end{cases} \tag{27.64}$$

where $\mathbf{B}^{LDA}$ is the usual topic-word matrix, $\mathbf{B}^{HMM}$ is the state-word HMM emission matrix and $\mathbf{A}^{HMM}$ is the state-state HMM transition matrix.

Inference in this model can be done with collapsed Gibbs sampling, analytically integrating out all the continuous quantities. See (Griffiths et al. 2004) for the details.

The results of applying this model (with $K = 200$ LDA topics and $C = 20$ HMM states) to the combined Brown and TASA corpora[4] are shown in Table 27.1. We see that the HMM generally is

4. The Brown corpus consists of 500 documents and 1,137,466 word tokens, with part-of-speech tags for each token.

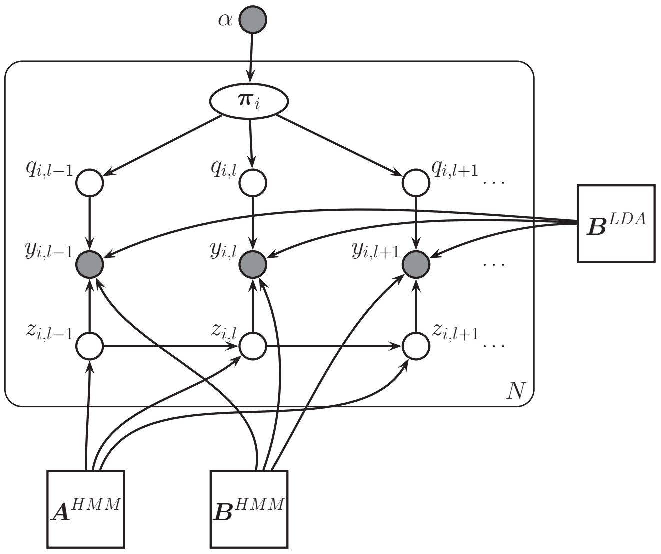


**Figure 27.13** LDA-HMM model.

1. In contrast to this approach, we study here how the overall network activity can $\boxed{\text{control}}$ single cell parameters such as input resistance, as well as time and space constants, parameters that are crucial for excitability and spariotemporal (sic) integration.

   The integrated architecture in this paper combines feed forward $\boxed{\text{control}}$ and error feedback adaptive $\boxed{\text{control}}$ using neural networks.

2. In other words, for our proof of convergence, we require the softassign algorithm to $\boxed{\text{return}}$ a doubly stochastic matrix as *sinkhorn theorem guarantees that it will instead of a matrix which is merely close to being doubly stochastic based on some reasonable metric.

   The aim is to construct a portfolio with a maximal expected $\boxed{\text{return}}$ for a given risk level and time horizon while simultaneously obeying *institutional or *legally required constraints.

3. The left $\boxed{\text{graph}}$ is the standard experiment the right from a training with # samples.

   The $\boxed{\text{graph}}$ $G$ is called the *guest $\boxed{\text{graph,}}$ and $H$ is called the host $\boxed{\text{graph.}}$

**Figure 27.14** Function and content words in the NIPS corpus, as distinguished by the LDA-HMM model. Graylevel indicates posterior probability of assignment to LDA component, with black being highest. The boxed word appears as a function word in one sentence, and as a content word in another sentence. Asterisked words had low frequency, and were treated as a single word type by the model. Source: Figure 4 of (Griffiths et al. 2004). Used with kind permission of Tom Griffiths.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| the | the | the | the | the | a | the | the | the |
| blood | , | , | of | a | the | , | , | , |
| , | and | and | , | of | of | of | a | a |
| of | of | of | to | , | , | a | of | in |
| body | a | in | in | in | in | and | and | game |
| heart | in | land | and | to | water | in | drink | ball |
| and | trees | to | classes | picture | is | story | alcohol | and |
| in | tree | farmers | government | film | and | is | to | team |
| to | with | for | a | image | matter | to | bottle | to |
| is | on | farm | state | lens | are | as | in | play |
| blood | forest | farmers | government | light | water | story | drugs | ball |
| heart | trees | land | state | eye | matter | stories | drug | game |
| pressure | forests | crops | federal | lens | molecules | poem | alcohol | team |
| body | land | farm | public | image | liquid | characters | people | * |
| lungs | soil | food | local | mirror | particles | poetry | drinking | baseball |
| oxygen | areas | people | act | eyes | gas | character | person | players |
| vessels | park | farming | states | glass | solid | author | effects | football |
| arteries | wildlife | wheat | national | object | substance | poems | marijuana | player |
| * | area | farms | laws | objects | temperature | life | body | field |
| breathing | rain | corn | department | lenses | changes | poet | use | basketball |
| the | in | he | * | be | said | can | time | , |
| a | for | it | new | have | made | would | way | ; |
| his | to | you | other | see | used | will | years | ( |
| this | on | they | first | make | came | could | day | : |
| their | with | i | same | do | went | may | part | ) |
| these | at | she | great | know | found | had | number | |
| your | by | we | good | get | called | must | kind | |
| her | from | there | small | go | | do | place | |
| my | as | this | little | take | | have | | |
| some | into | who | old | find | | did | | |

**Table 27.1** Upper row: Topics extracted by the LDA model when trained on the combined Brown and TASA corpora. Middle row: topics extracted by LDA part of LDA-HMM model. Bottom row: topics extracted by HMM part of LDA-HMM model. Each column represents a single topic/class, and words appear in order of probability in that topic/class. Since some classes give almost all probability to only a few words, a list is terminated when the words account for 90% of the probability mass. Source: Figure 2 of (Griffiths et al. 2004). Used with kind permission of Tom Griffiths.

responsible for syntactic words, and the LDA for semantics words. If we did not have the HMM, the LDA topics would get "polluted" by function words (see top of figure), which is why such words are normally removed during preprocessing.

The model can also help disambiguate when the same word is being used syntactically or semantically. Figure 27.14 shows some examples when the model was applied to the NIPS corpus.[5] We see that the roles of words are distinguished, e.g., "we require the algorithm to *return* a matrix" (verb) vs "the maximal expected *return*" (noun). In principle, a part of speech tagger could disambiguate these two uses, but note that (1) the LDA-HMM method is fully unsupervised (no POS tags were used), and (2) sometimes a word can have the same POS tag, but different senses, e.g., "the left graph" (a synactic role) vs "the graph $G$" (a semantic role).

The topic of probabilistic models for syntax and semantics is a vast one, which we do not

The TASA corpus is an untagged collection of educational materials consisting of 37,651 documents and 12,190,931 word tokens. Words appearing in fewer than 5 documents were replaced with an asterisk, but punctuation was included. The combined vocabulary was of size 37,202 unique words.

5. NIPS stands for "Neural Information Processing Systems". It is one of the top machine learning conferences. The NIPS corpus volumes 1–12 contains 1713 documents.

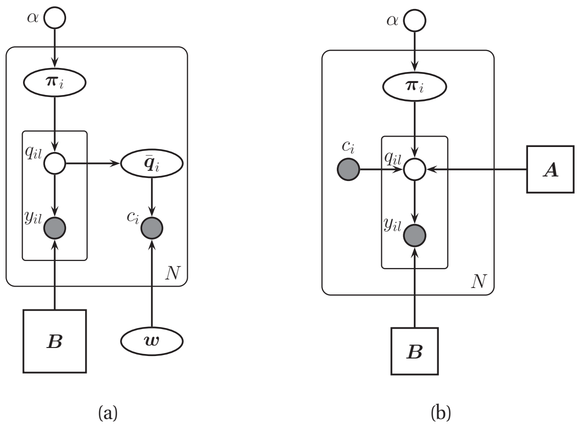


**Figure 27.15** (a) Supervised LDA. (b) Discriminative LDA.

have space to delve into any more. See e.g., (Jurafsky and Martin 2008) for further information.

### 27.4.4 Supervised LDA

In this section, we discuss extensions of LDA to handle side information of various kinds beyond just words.

#### 27.4.4.1 Generative supervised LDA

Suppose we have a variable length sequence of words $y_{il} \in \{1, \ldots, V\}$ as usual, but we also have a class label $c_i \in \{1, \ldots, C\}$. How can we predict $c_i$ from $\mathbf{y}_i$? There are many possible approaches, but most are direct mappings from the words to the class. In some cases, such as **sentiment analysis**, we can get better performance by first performing inference, to try to disambiguate the meaning of words. For example, suppose the goal is to determine if a document is a favorable review of a movie or not. If we encounter the phrase “Brad Pitt was excellent until the middle of the movie”, the word “excellent” may lead us to think the review is positive, but clearly the overall sentiment is negative.

One way to tackle such problems is to build a joint model of the form $p(c_i, \mathbf{y}_i|\boldsymbol{\theta})$. (Blei and McAuliffe 2010) proposes an approach, called **supervised LDA**, where the class label $c_i$ is generated from the topics as follows:

$$p(c_i|\overline{\mathbf{q}}_i) = \text{Ber}(\text{sigm}(\mathbf{w}^T\overline{\mathbf{q}}_i)) \tag{27.65}$$

Here $\overline{\mathbf{q}}_i$ is the empirical topic distribution for document $i$:

$$\overline{q}_{ik} \triangleq \frac{1}{L_i} \sum_{i=1}^{L_i} q_{ilk} \tag{27.66}$$

See Figure 27.15(a) for an illustration.

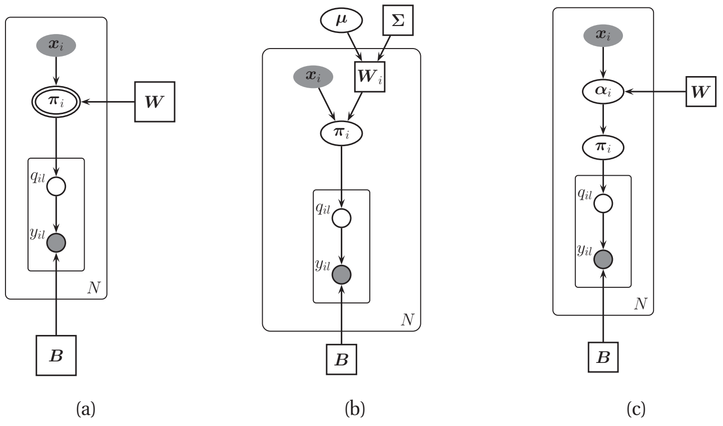


**Figure 27.16** Discriminative variants of LDA. (a) Mixture of experts aka MR-LDA. The double ring denotes a node that $\boldsymbol{\pi}_i$ a deterministic function of its parents. (b) Mixture of experts with random effects. (c) DMR-LDA.

We can fit this model using Monte Carlo EM: run the collapsed Gibbs sampler in the E step, to compute $\mathbb{E}[\overline{q}_{ik}]$, and then use this as the input feature to a standard logistic regression package.

#### 27.4.4.2 Discriminative supervised LDA

An alternative approach, known as **discriminative LDA** (Lacoste-Julien et al. 2009), is shown in Figure 27.15(b). This is a discriminative model of the form $p(\mathbf{y}_i|c_i, \boldsymbol{\theta})$. The only change from regular LDA is that the topic prior becomes input dependent, as follows:

$$p(q_{il}|\boldsymbol{\pi}_i, c_i = c, \boldsymbol{\theta}) = \text{Cat}(\mathbf{A}_c \boldsymbol{\pi}) \tag{27.67}$$

where $\mathbf{A}_c$ is a $K \times K$ stochastic matrix.

So far, we have assumed the "side information" is a single categorical variable $c_i$. Often we have high dimensional covariates $\mathbf{x}_i \in \mathbb{R}^D$. For example, consider the task of **image tagging**. The idea is that $y_{il}$ represent correlated tags or labels, which we want to predict given $\mathbf{x}_i$. We now discuss several attempts to extend LDA so that it can generate tags given the inputs.

The simplest approach is to use a mixture of experts (Section 11.2.4) with multiple outputs. This is just like LDA except we replace the Dirichlet prior on $\boldsymbol{\pi}_i$ with a deterministic function of the input:

$$\boldsymbol{\pi}_i \quad = \quad \mathcal{S}(\mathbf{W}\mathbf{x}_i) \tag{27.68}$$

In (Law et al. 2010), this is called **multinomial regression LDA**. See Figure 27.16(a). Eliminating the deterministic $\boldsymbol{\pi}_i$ we have

$$p(q_{il}|\mathbf{x}_i, \mathbf{W}) = \text{Cat}(\mathcal{S}(\mathbf{W}\mathbf{x}_i)) \tag{27.69}$$

We can fit this with EM in the usual way. However, (Law et al. 2010) suggest an alternative. First fit an unsupervised LDA model based only on $\mathbf{y}_i$; then treat the inferred $\boldsymbol{\pi}_i$ as data, and fit a multinomial logistic regression model mapping $\mathbf{x}_i$ to $\boldsymbol{\pi}_i$. Although this is fast, fitting LDA in an unsupervised fashion does not necessarily result in a discriminative set of latent variables, as discussed in (Blei and McAuliffe 2010).

There is a more subtle problem with this model. Since $\boldsymbol{\pi}_i$ is a deterministic function of the inputs, it is effectively observed, rendering the $q_{il}$ (and hence the tags $y_{il}$) independent. In other words,

$$p(\mathbf{y}_i|\mathbf{x}_i,\boldsymbol{\theta}) = \prod_{l=1}^{L_i} p(y_{il}|\mathbf{x}_i,\boldsymbol{\theta}) = \prod_{l=1}^{L_i}\sum_k p(y_{il}|q_{il}=k,\mathbf{B})p(q_{il}=k|\mathbf{x}_i,\mathbf{W}) \tag{27.70}$$

This means that if we observe the value of one tag, it will have no influence on any of the others. This may explain why the results in (Law et al. 2010) only show negligible improvement over predicting each tag independently.

One way to induce correlations is to make $\mathbf{W}$ a random variable. The resulting model is shown in Figure 27.16(b). We call this a **random effects mixture of experts**. We typically assume a Gaussian prior on $\mathbf{W}_i$. If $\mathbf{x}_i = 1$, then $p(q_{il}|\mathbf{x}_i,\mathbf{w}_i) = \text{Cat}(\mathcal{S}(\mathbf{w}_i))$, so we recover the correlated topic model. It is possible to extend this model by adding Markovian dynamics to the $q_{il}$ variables. This is called a **conditional topic random field** (Zhu and Xing 2010).

A closely related approach, known as **Dirichlet multinomial regression LDA** (Mimno and McCallum 2008), is shown in Figure 27.16(c). This is identical to standard LDA except we make $\boldsymbol{\alpha}$ a function of the input

$$\boldsymbol{\alpha}_i = \exp(\mathbf{W}\mathbf{x}_i) \tag{27.71}$$

where $\mathbf{W}$ is a $K \times D$ matrix. Eliminating the deterministic $\boldsymbol{\alpha}_i$ we have

$$\boldsymbol{\pi}_i \sim \text{Dir}(\exp(\mathbf{W}\mathbf{x}_i)) \tag{27.72}$$

Unlike (Law et al. 2010), this model allows information to flow between tags via the latent $\boldsymbol{\pi}_i$.

A variant of this model, where $\mathbf{x}_i$ corresponds to a bag of discrete labels and $\boldsymbol{\pi}_i \sim \text{Dir}(\boldsymbol{\alpha} \odot \mathbf{x}_i)$, is known as **labeled LDA** (Ramage et al. 2009). In this case, the labels $\mathbf{x}_i$ are in 1:1 correspondence with the latent topics, which makes the resulting topics much more interpretable. An extension, known as **partially labeled LDA** (Ramage et al. 2011), allows each label to have multiple latent sub-topics; this model includes LDA, labeled LDA and a multinomial mixture model as special cases.

#### 27.4.4.3 Discriminative categorical PCA

An alternative to using LDA is to expand the categorical PCA model with inputs, as shown in Figure 27.17(a). Since the latent space is now real-valued, we can use simple linear regression for the input-hidden mapping. For the hidden-output mapping, we use traditional catPCA:

$$p(\mathbf{z}_i|\mathbf{x}_i,\mathbf{V}) = \mathcal{N}(\mathbf{V}\mathbf{x}_i,\boldsymbol{\Sigma}) \tag{27.73}$$

$$p(\mathbf{y}_i|\mathbf{z}_i,\mathbf{W}) = \prod_l \text{Cat}(y_{il}|\mathcal{S}(\mathbf{W}\mathbf{z}_i)) \tag{27.74}$$

This model is essentially a probabilistic neural network with one hidden layer, as shown in Figure 27.17(b), but with exchangeable output (e.g., to handle variable numbers of tags). The

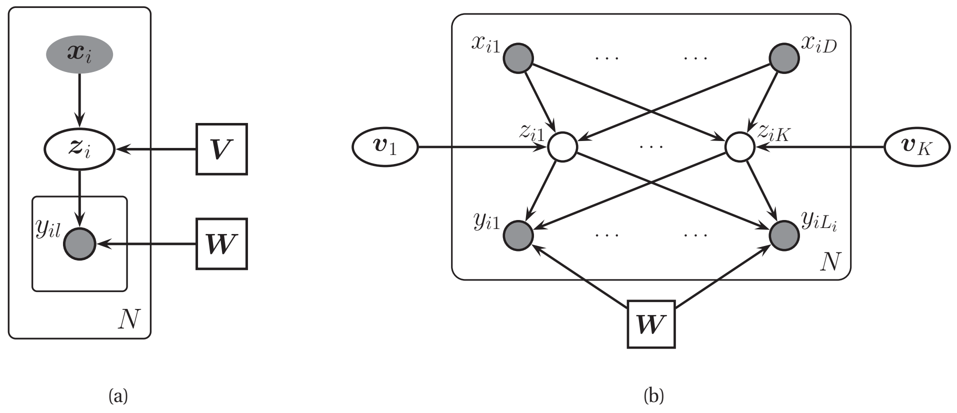


**Figure 27.17** (a) Categorical PCA with inputs and exchangeable outputs. (b) Same as (a), but with the vector nodes expanded out.

key difference from a neural net is that information can flow between the $y_{il}$'s via the latent **bottleneck layer** $\mathbf{z}_i$. This should work better than a conventional neural net when the output labels are highly correlated, even after conditioning on the features; this problem frequently arises in multi label classification. Note that we could allow a direct $\mathbf{x}_i$ to $\mathbf{y}_i$ arc, but this would require too many parameters if the number of labels is large.[6]

We can fit this model with a small modification of the variational EM algorithm in Section 12.4. If we use this model for regression, rather than classification, we can perform the E step exactly, by modifying the EM algorithm for factor analysis. (Ma et al. 1997) reports that this method converges faster than standard backpropagation.

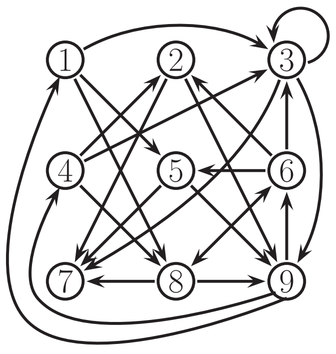
We can also extend the model so that the prior on $\mathbf{z}_i$ is a mixture of Gaussians using input-dependent means. If the output is Gaussian, this corresponds to a mixture of discriminative factor analysers (Fokoue 2005; Zhou and Liu 2008). If the output is categorical, this would be an (as yet unpublished) model, which we could call "discriminative mixtures of categorical factor analyzers".

## 27.5 LVMs for graph-structured data

Another source of discrete data is when modeling graph or network structures. To see the connection, recall that any graph on $D$ nodes can be represented as a $D \times D$ **adjacency matrix G**, where $G(i,j) = 1$ iff there is an edge from node $i$ to node $j$. Such matrices are binary, and often very sparse. See Figure 27.19 for an example.

Graphs arise in many application areas, such as modeling social networks, protein-protein interaction networks, or patterns of disease transmission between people or animals. There are usually two primary goals when analysing such data: first, try to discover some "interesting

6. A non-probabilistic version of this idea, using squared loss, was proposed in (Ji et al. 2010). This is similar to a linear feed-forward neural network with an additional edge from $\mathbf{x}_i$ directly to $\mathbf{y}_i$.



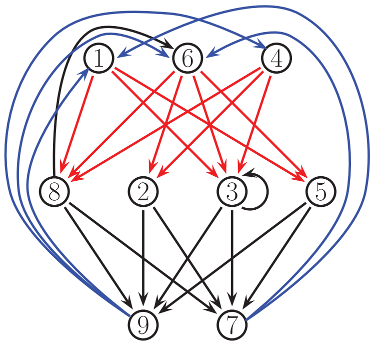
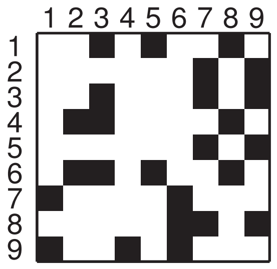


**Figure 27.18** (a) A directed graph. (b) The same graph, with the nodes partitioned into 3 groups, making the block structure more apparent.



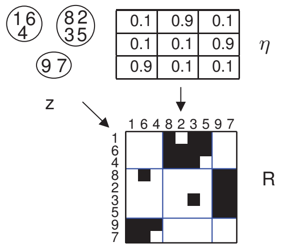


**Figure 27.19** (a) Adjacency matrix for the graph in Figure 27.18(a). (b) Rows and columns are shown permuted to show the block structure. We also sketch of how the stochastic block model can generate this graph. From Figure 1 of (Kemp et al. 2006). Used with kind permission of Charles Kemp.

structure" in the graph, such as clusters or communities; second, try to predict which links might occur in the future (e.g., who will make friends with whom). Below we summarize some models that have been proposed for these tasks, some of which are related to LDA. Futher details on these and other approaches can be found in e.g., (Goldenberg et al. 2009) and the references therein.

### 27.5.1 Stochastic block model

In Figure 27.18(a) we show a directed graph on 9 nodes. There is no apparent structure. However, if we look more deeply, we see it is possible to partition the nodes into three groups or blocks, $B_1 = \{1, 4, 6\}$, $B_2 = \{2, 3, 5, 8\}$, and $B_3 = \{7, 9\}$, such that most of the connections go from nodes in $B_1$ to $B_2$, or from $B_2$ to $B_3$, or from $B_3$ to $B_1$. This is illustrated in Figure 27.18(b).

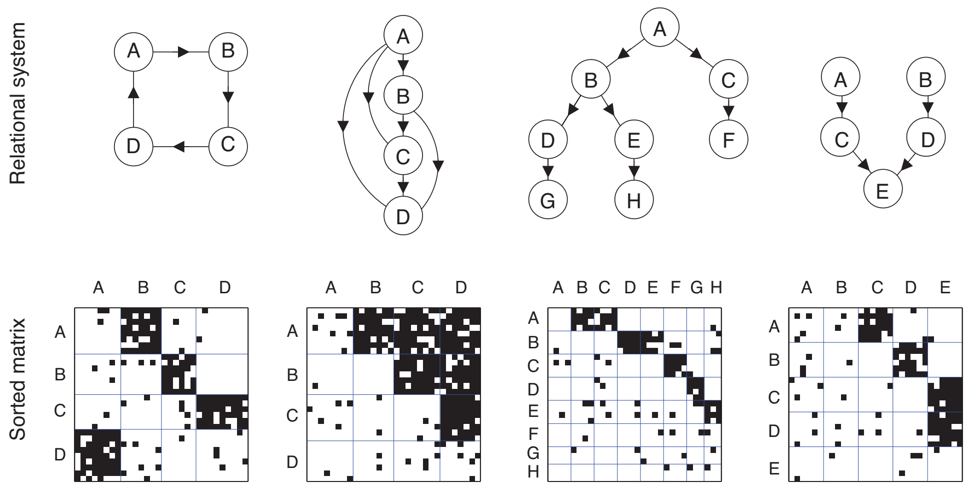


**Figure 27.20** Some examples of graphs generated using the stochastic block model with different kinds of connectivity patterns between the blocks. The abstract graph (between blocks) represent a ring, a dominance hierarchy, a common-cause structure, and a common-effect structure. From Figure 4 of (Kemp et al. 2010). Used with kind permission of Charles Kemp.

The problem is easier to understand if we plot the adjacency matrices. Figure 27.19(a) shows the matrix for the graph with the nodes in their original ordering. Figure 27.19(b) shows the matrix for the graph with the nodes in their permtuted ordering. It is clear that there is block structure.

We can make a generative model of block structured graphs as follows. First, for every node, sample a latent block $q_i \sim \mathrm{Cat}(\boldsymbol{\pi})$, where $\pi_k$ is the probability of choosing block $k$, for $k = 1 : K$. Second, choose the probability of connecting group $a$ to group $b$, for all pairs of groups; let us denote this probability by $\eta_{a,b}$. This can come from a beta prior. Finally, generate each edge $R_{ij}$ using the following model:

$$p(R_{ij} = r | q_i = a, q_j = b, \boldsymbol{\eta}) = \mathrm{Ber}(r | \eta_{a,b}) \quad (27.75)$$

This is called the **stochastic block model** (Nowicki and Snijders 2001). Figure 27.21(a) illustrates the model as a DGM, and Figure 27.19(c) illustrates how this model can be used to cluster the nodes in our example.

Note that this is quite different from a conventional clustering problem. For example, we see that all the nodes in block 3 are grouped together, even though there are no connections between them. What they share is the property that they "like to" connect to nodes in block 1, and to receive connections from nodes in block 2. Figure 27.20 illustrates the power of the model for generating many different kinds of graph structure. For example, some social networks have hierarchical structure, which can be modeled by clustering people into different social strata, whereas others consist of a set of cliques.

Unlike a standard mixture model, it is not possible to fit this model using exact EM, because all the latent $q_i$ variables become correlated. However, one can use variational EM (Airoldi et al.

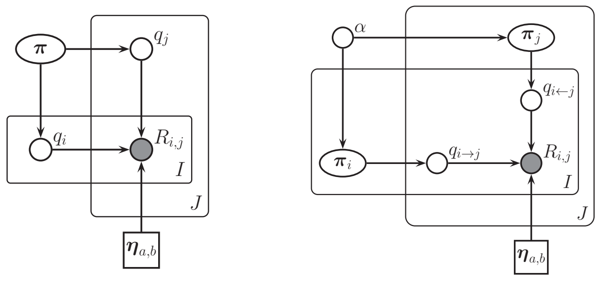


**Figure 27.21** (a) Stochastic block model. (b) Mixed membership stochastic block model.

2008), collapsed Gibbs sampling (Kemp et al. 2006), etc. We omit the details (which are similar to the LDA case).

In (Kemp et al. 2006), they lifted the restriction that the number of blocks $K$ be fixed, by replacing the Dirichlet prior on $\boldsymbol{\pi}$ by a Dirichlet process (see Section 25.2.2). This is known as the infinite relational model. See Section 27.6.1 for details.

If we have features associated with each node, we can make a discriminative version of this model, for example by defining

$$p(R_{ij} = r|q_i = a, q_j = b, \mathbf{x}_i, \mathbf{x}_j, \boldsymbol{\theta}) = \text{Ber}(r|\mathbf{w}_{a,b}^T f(\mathbf{x}_i, \mathbf{x}_j)) \tag{27.76}$$

where $f(\mathbf{x}_i, \mathbf{x}_j)$ is some way of combining the feature vectors. For example, we could use concatenation, $[\mathbf{x}_i, \mathbf{x}_j]$, or elementwise product $\mathbf{x}_i \otimes \mathbf{x}_j$ as in supervised LDA. The overall model is like a relational extension of the mixture of experts model.

## 27.5.2 Mixed membership stochastic block model

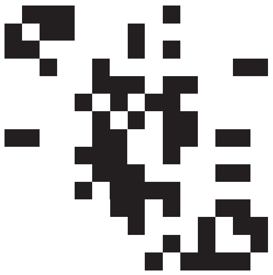
In (Airoldi et al. 2008), they lifted the restriction that each node only belong to one cluster. That is, they replaced $q_i \in \{1, \ldots, K\}$ with $\boldsymbol{\pi}_i \in S_K$. This is known as the **mixed membership stochastic block model**, and is similar in spirit to **fuzzy clustering** or **soft clustering**. Note that $\pi_{ik}$ is not the same as $p(z_i = k|\mathcal{D})$; the former represents **ontological uncertainty** (to what degree does each object belong to a cluster) wheras the latter represents **epistemological uncertainty** (which cluster does an object belong to). If we want to combine epistemological and ontological uncertainty, we can compute $p(\boldsymbol{\pi}_i|\mathcal{D})$.

In more detail, the generative process is as follows. First, each node picks a distribution over blocks, $\boldsymbol{\pi}_i \sim \text{Dir}(\boldsymbol{\alpha})$. Second, choose the probability of connecting group $a$ to group $b$, for all pairs of groups, $\eta_{a,b} \sim \beta(\alpha, \beta)$. Third, for each edge, sample two discrete variables, one for each direction:

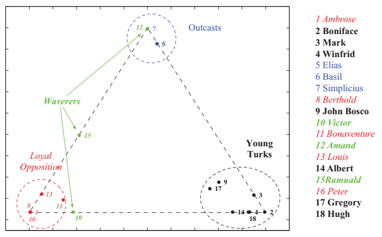
$$q_{i\rightarrow j} \sim \text{Cat}(\boldsymbol{\pi}_i), \; q_{i\leftarrow j} \sim \text{Cat}(\boldsymbol{\pi}_j) \tag{27.77}$$

Finally, generate each edge $R_{ij}$ using the following model:

$$p(R_{ij} = 1|q_{i\rightarrow j} = a, q_{i\leftarrow j} = b, \boldsymbol{\eta}) = \eta_{a,b} \tag{27.78}$$



(a)



(b)

**Figure 27.22** (a) Who-likes-whom graph for Sampson's monks. (b) Mixed membership of each monk in one of three groups. From Figures 2-3 of (Airoldi et al. 2008). Used with kind permission of Edo Airoldi.

See Figure 27.21(b) for the DGM.

Unlike the regular stochastic block model, each node can play a different role, depending on who it is connecting to. As an illustration of this, we will consider a data set that is widely used in the social networks analysis literature. The data concerns who-likes-whom amongst of group of 18 monks. It was collected by hand in 1968 by Sampson (Sampson 1968) over a period of months. (These days, in the era of social media such as Facebook, a social network with only 18 people is trivially small, but the methods we are discussing can be made to scale.) Figure 27.22(a) plots the raw data, and Figure 27.22(b) plots $\mathbb{E}\left[\boldsymbol{\pi}\right]_i$ for each monk, where $K = 3$. We see that most of the monk belong to one of the three clusters, known as the "young turks", the "outcasts" and the "loyal opposition". However, some individuals, notably monk 15, belong to two clusters; Sampson called these monks the "waverers". It is interesting to see that the model can recover the same kinds of insights as Sampson derived by hand.

One prevalent problem in social network analysis is missing data. For example, if $R_{ij} = 0$, it may be due to the fact that person $i$ and $j$ have not had an opportunity to interact, or that data is not available for that interaction, as opposed to the fact that these people don't want to interact. In other words, *absence of evidence is not evidence of absence*. We can model this by modifying the observation model so that with probability $\rho$, we generate a 0 from the background model, and we only force the model to explain observed 0s with probability $1 - \rho$. In other words, we robustify the observation model to allow for outliers, as follows:

$$p(R_{ij} = r | q_{i \to j} = a, q_{i \leftarrow j} = b, \boldsymbol{\eta}) = \rho \delta_0(r) + (1 - \rho)\text{Ber}(r | \eta_{a,b}) \tag{27.79}$$

See (Airoldi et al. 2008) for details.

## 27.5.3 Relational topic model

In many cases, the nodes in our network have atttributes. For example, if the nodes represent academic papers, and the edges represent citations, then the attributes include the text of the document itself. It is therefore desirable to create a model that can explain the text and the link structure concurrently. Such a model can predict links given text, or even vice versa.

The **relational topic model** (RTM) (Chang and Blei 2010) is one way to do this. This is a

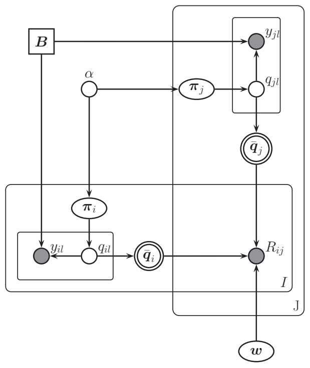


**Figure 27.23** DGM for the relational topic model.

simple extension of supervised LDA (Section 27.4.4.1), where the response variable $R_{ij}$ (which represents whether there is an edge between nodes $i$ and $j$) is modeled as follows:

$$p(R_{ij} = 1|\overline{\mathbf{q}}_i, \overline{\mathbf{q}}_j, \boldsymbol{\theta}) = \text{sigm}(\mathbf{w}^T(\overline{\mathbf{q}}_i \otimes \overline{\mathbf{q}}_j) + w_0) \tag{27.80}$$

Recall that $\overline{\mathbf{q}}_i$ is the empirical topic distribution for document $i$, $\overline{q}_{ik} \triangleq \frac{1}{L_i}\sum_{i=1}^{L_i} q_{ilk}$. See Figure 27.23

Note that it is important that $R_{ij}$ depend on the actual topics chosen, $\overline{\mathbf{q}}_i$ and $\overline{\mathbf{q}}_j$, and not on the topic distributions, $\boldsymbol{\pi}_i$ and $\boldsymbol{\pi}_j$, otherwise predictive performance is not as good. The intuitive reason for this is as follows: if $R_{ij}$ is a child of $\boldsymbol{\pi}_i$ and $\boldsymbol{\pi}_j$, it will be treated as just another word, similar to the $q_{il}$'s and $y_{il}$'s; but since there are many more words than edges, the graph structure information will get “washed out”. By making $R_{ij}$ a child of $\overline{\mathbf{q}}_i$ and $\overline{\mathbf{q}}_j$, the graph information can influence the choice of topics more directly.

One can fit this model in a manner similar to SLDA. See (Chang and Blei 2010) for details. The method does better at predicting missing links than the simpler approach of first fitting an LDA model, and then using the $\overline{\mathbf{q}}_i$'s as inputs to a logistic regression problem. The reason is analogous to the superiority of partial least squares (Section 12.5.2) to PCA+ linear regression, namely that the RTM learns a latent space that is forced to be predictive of the graph structure and words, whereas LDA might learn a latent space that is not useful for predicting the graph.

## 27.6 LVMs for relational data

Graphs can be used to represent data which represents the relation amongst variables of a certain type, e.g., friendship relationships between people. But often we have multiple types of objects, and multiple types of relations. For example, Figure 27.24 illustrates two relations, one between people and people, and one between people and movies.

In general, we define a $k$-ary **relation** $R$ as a subset of $k$-tuples of the appropriate types:

$$R \subseteq T_1 \times T_2 \times \cdots \times T_k \tag{27.81}$$

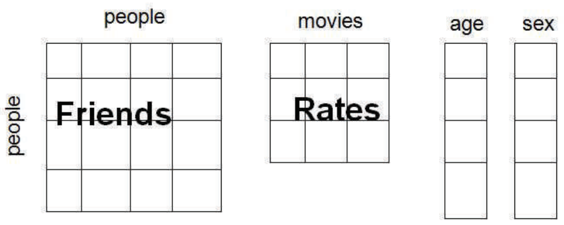


**Figure 27.24** Example of relational data. There are two types of objects, *people* and *movies*; one 2-ary relation, *friends*: *people* $\times$ *people* $\to \{0,1\}$ and one 2-ary function, *rates*: *people* $\times$ *movie* $\to \mathbb{R}$. *Age* and *sex* are attributes (unary functions) of the *people* class.

where $T_i$ are sets or types. A binary, pairwise or dyadic relation is a relation defined on pairs of objects. For example, the *seen* relation between people and movies might be represented as the set of movies that people have seen. We can either represent this explicitly as a set, such as

```
seen  = { (Bob, StarWars), (Bob, TombRaider), (Alice, Jaws)}
```

or implicitly, using an indicator function for the set:

```
seen(Bob, StarWars)=1, seen(Bob, TombRaider)=1, seen(Alice, Jaws)=1
```

A relation between two entities of types $T^1$ and $T^2$ can be represented as a binary function $R : T^1 \times T^2 \to \{0,1\}$, and hence as a binary matrix. This can also be represented as a bipartite graph, in which we have nodes of two types. If $T^1 = T^2$, this becomes a regular directed graph, as in Section 27.5. However, there are some situations that are not so easily modelled by graphs, but which can still be modelled by relations. For example, we might have a ternary relation, $R : T^1 \times T^1 \times T^2 \to \{0,1\}$, where, say, $R(i,j,k) = 1$ iff protein $i$ interacts with protein $j$ when chemical $k$ is present. This can be modelled by a 3d binary matrix. We will give some examples of this in Section 27.6.1.

Making probabilistic models of relational data is called **statistical relational learning** (Getoor and Taskar 2007). One approach is to directly model the relationship between the variables using graphical models; this is known as **probabilistic relational modeling**. Another approach is to use latent variable models, as we discuss below.

## 27.6.1 Infinite relational model

It is straightforward to extend the stochastic block model to model relational data: we just associate a latent variable $q_i^t \in \{1, \ldots, K_t\}$ with each entity $i$ of each type $t$. We then define the probability of the relation holding between specific entities by looking up the probability of the relation holding between entities of that type. For example, if $R : T^1 \times T^1 \times T^2 \to \{0,1\}$, we have

$$p(R(i,j,k) = 1 | q_i^1 = a, q_j^1 = b, q_k^2 = c, \boldsymbol{\eta}) = \eta_{a,b,c} \tag{27.82}$$

If we allow the number of clusters $K_t$ for each type to be unbounded, by using a Dirichlet process, the model is called the **infinite relational model** (IRM) (Kemp et al. 2006). An essentially

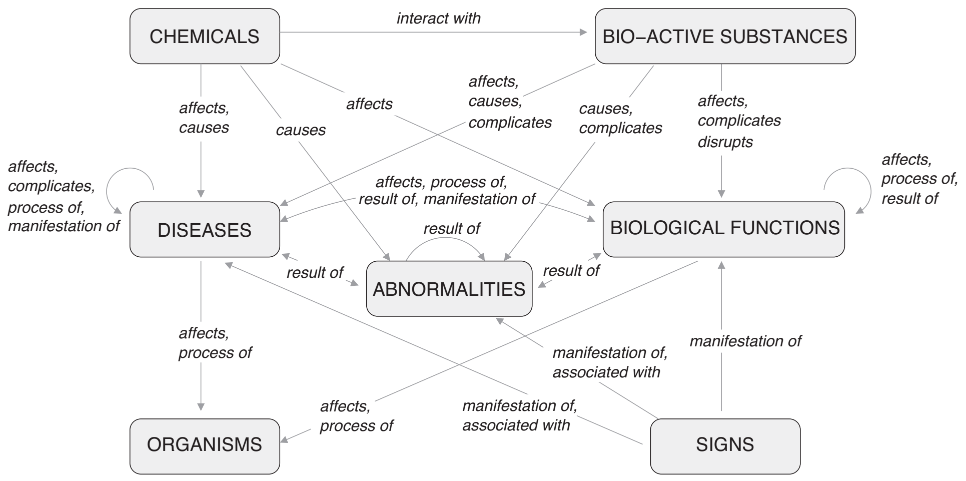

CHEMICALS
BIO–ACTIVE SUBSTANCES
DISEASES
ABNORMALITIES
BIOLOGICAL FUNCTIONS
ORGANISMS
SIGNS
interact with
affects, causes
causes
affects
affects, causes, complicates
causes, complicates
affects, complicates disrupts
affects, complicates, process of, manifestation of
affects, process of, result of, manifestation of
result of
affects, process of, result of
result of
result of
affects, process of
manifestation of, associated with
manifestation of
affects, process of
manifestation of, associated with

**Figure 27.25** Illustration of an ontology learned by IRM applied to the Unified Medical Language System. The boxes represent 7 of the 14 concept clusters. Predicates that belong to the same cluster are grouped together, and associated with edges to which they pertain. All links with weight above 0.8 have been included. From Figure 9 of (Kemp et al. 2010). Used with kind permission of Charles Kemp.

identical model, under the name **infinite hidden relational model** (IHRM), was concurrently proposed in (Xu et al. 2006). We can fit this model with variational Bayes (Xu et al. 2006, 2007) or collapsed Gibbs sampling (Kemp et al. 2006). Rather than go into algorithmic detail, we just sketch some interesting applications.

#### 27.6.1.1 Learning ontologies

An **ontology** refers to an organisation of knowledge. In AI, ontologies are often built by hand (see e.g., (Russell and Norvig 2010)), but it is interesting to try and learn them from data. In (Kemp et al. 2006), they show how this can be done using the IRM.

The data comes from the Unified Medical Language System (McCray 2003), which defines a semantic network with 135 concepts (such as "disease or syndrome", "diagnostic procedure", "animal"), and 49 binary predicates (such as "affects", "prevents"). We can represent this as a ternary relation $R : T^1 \times T^1 \times T^2 \to \{0, 1\}$, where $T^1$ is the set of concepts and $T^2$ is the set of binary predicates. The result is a 3d cube. We can then apply the IRM to partition the cube into regions of roughly homogoneous response. The system found 14 concept clusters and 21 predicate clusters. Some of these are shown in Figure 27.25. The system learns, for example, that biological functions affect organisms (since $\eta_{a,b,c} \approx 1$ where $a$ represents the biological function cluster, $b$ represents the organism cluster, and $c$ represents the affects cluster).

#### 27.6.1.2 Clustering based on relations and features

We can also use IRM to cluster objects based on their relations and their features. For example, (Kemp et al. 2006) consider a political dataset (from 1965) consisting of 14 countries, 54 binary

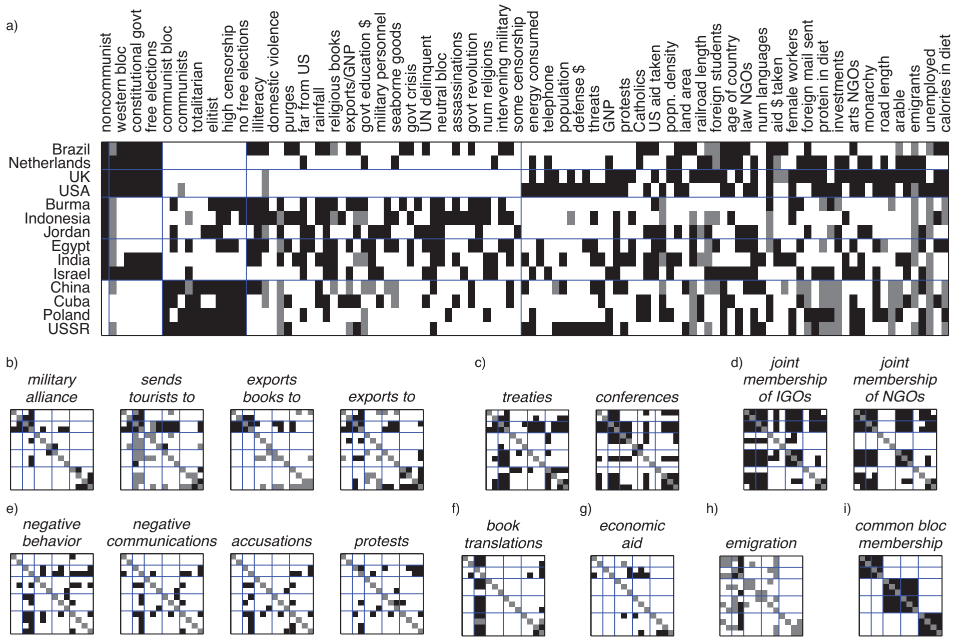


**Figure 27.26** Illustration of IRM applied to some political data containing features and pairwise interactions. Top row (a). the partition of the countries into 5 clusters and the features into 5 clusters. Every second column is labelled with the name of the corresponding feature. Small squares at bottom (a-i): these are 8 of the 18 clusters of interaction types. From Figure 6 of (Kemp et al. 2006). Used with kind permission of Charles Kemp.

predicates representing interaction types between countries (e.g., "sends tourists to", "economic aid"), and 90 features (e.g., "communist", "monarchy"). To create a binary dataset, real-valued features were thresholded at their mean, and categorical variables were dummy-encoded. The data has 3 types: $T^1$ represents countries, $T^2$ represents interactions, and $T^3$ represents features. We have two relations: $R^1 : T^1 \times T^1 \times T^2 \rightarrow \{0, 1\}$, and $R^2 : T^1 \times T^3 \rightarrow \{0, 1\}$. (This problem therefore combines aspects of both the biclustering model and the ontology discovery model.) When given multiple relations, the IRM treats them as conditionally independent. In this case, we have

$$p(\mathbf{R}^1, \mathbf{R}^2, \mathbf{q}^1, \mathbf{q}^2, \mathbf{q}^3|\boldsymbol{\theta}) = p(\mathbf{R}^1|\mathbf{q}^1, \mathbf{q}^2, \boldsymbol{\theta})p(\mathbf{R}^2|\mathbf{q}^1, \mathbf{q}^3, \boldsymbol{\theta}) \tag{27.83}$$

The results are shown in Figure 27.26. The IRM divides the 90 features into 5 clusters, the first of which contains "noncommunist", which captures one of the most important aspects of this Cold-War era dataset. It also clusters the 14 countries into 5 clusters, reflecting natural geo-political groupings (e.g., US and UK, or the Communist Bloc), and the 54 predicates into 18 clusters, reflecting similar relationships (e.g., "negative behavior and "accusations").

### 27.6.2 Probabilistic matrix factorization for collaborative filtering

As discussed in Section 1.3.4.2, collaborative filtering (CF) requires predicting entries in a matrix $R : T^1 \times T^2 \to \mathbb{R}$, where for example $R(i, j)$ is the rating that user $i$ gave to movie $j$. Thus we see that CF is a kind of relational learning problem (and one with particular commercial importance).

Much of the work in this area makes use of the data that Netflix made available in their competition. In particular, a large 17,770 $\times$ 480,189 movie x user ratings matrix is provided. The full matrix would have $\sim 8.6 \times 10^9$ entries, but only 100,480,507 (about 1%) of the entries are observed, so the matrix is extremely sparse. In addition the data is quite imbalanced, with many users rating fewer than 5 movies, and a few users rating over 10,000 movies. The validation set is 1,408,395 (movie,user) pairs. Finally, there is a separate test set with 2,817,131 (movie,user) pairs, for which the ranking is known but withheld from contestants. The performance measure is root mean square error:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X(m_i, u_i) - \hat{X}(m_i, u_i))^2} \tag{27.84}$$

where $X(m_i, u_i)$ is the true rating of user $u_i$ on movie $m_i$, and $\hat{X}(m_i, u_i)$ is the prediction. The baseline system, known as Cinematch, had an RMSE on the training set of 0.9514, and on the test set of 0.9525. To qualify for the grand prize, teams needed to reduce the test RMSE by 10%, i.e., get a test RMSE of 0.8563 or less. We will discuss some of the basic methods used byt the winning team below.

Since the ratings are drawn from the set $\{0, 1, 2, 3, 4, 5\}$, it is tempting to use a categorical observation model. However, this does not capture the fact that the ratings are ordered. Although we could use an ordinal observation model, in practice people use a Gaussian observation model for simplicity. One way to make the model better match the data is to pass the model's predicted mean response through a sigmoid, and then to map the $[0, 1]$ interval to $[0, 5]$ (Salakhutdinov and Mnih 2008). Alternatively we can make the data a better match to the Gaussian model by transforming the data using $R_{ij} = \sqrt{6 - R_{ij}}$ (Aggarwal and Merugu 2007).

We could use the IRM for the CF task, by associating a discrete latent variable for each user $q_i^u$ and for each movie or video $q_j^v$, and then defining

$$p(R_{ij} = r | q_i^u = a, q_j^v = b, \boldsymbol{\theta}) = \mathcal{N}(r | \mu_{a,b}, \sigma^2) \tag{27.85}$$

This is just another example of co-clustering. We can also extend the model to generate side information, such as attributes about each user and/or movie. See Figure 27.27 for an illustration.

Another possibility is to replace the discrete latent variables with continuous latent variables $\boldsymbol{\pi}_i^u \in S_{K_u}$ and $\boldsymbol{\pi}_j^v \in S_{K_v}$. However, it has been found (see e.g., (Shan and Banerjee 2010)) that one obtains much better results by using unconstrained real-valued latent factors for each user $\mathbf{u}_i \in \mathbb{R}^K$ and each movie $\mathbf{v}_j \in \mathbb{R}^K$.[7] We then use a likelihood of the form

$$p(R_{ij} = r | \mathbf{u}_i, \mathbf{v}_j) = \mathcal{N}(r | \mathbf{u}_i^T \mathbf{v}_j, \sigma^2) \tag{27.86}$$

7. Good results with discrete latent variables have been obtained on some datasets that are smaller than Netflix, such as MovieLens and EachMovie. However, these datasets are much easier to predict, because there is less imbalance between the number of reviews performed by different users (in Netflix, some users have rated more than 10,000 movies, whereas others have rated less than 5).

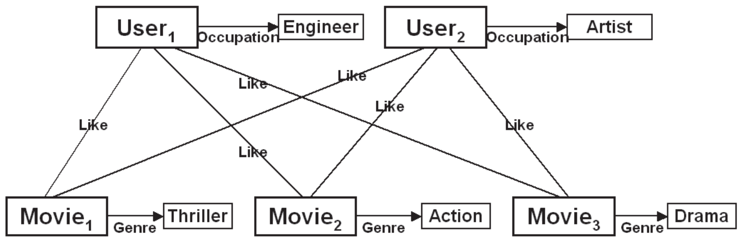


**Figure 27.27** Visualization of a small relational dataset, where we have one relation, likes(user, movie), and features for movies (here, genre) and users (here, occupation). From Figure 5 of (Xu et al. 2008). Used with kind permission of Zhao Xu.

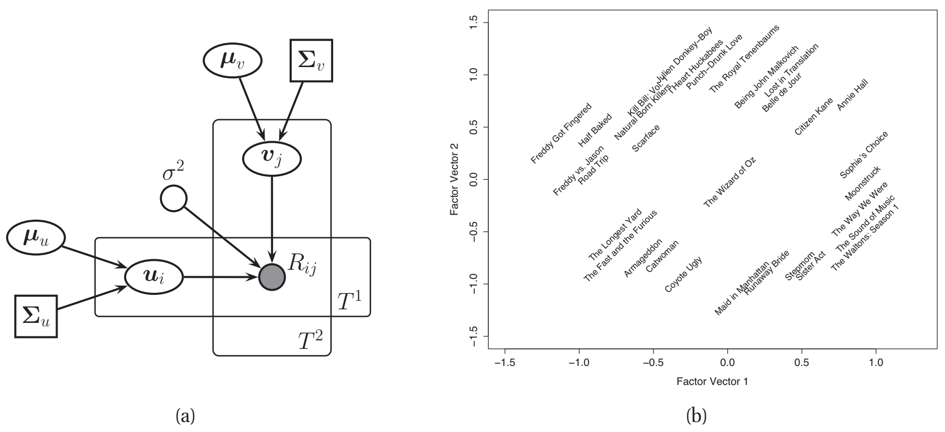


**Figure 27.28** (a) A DGM for probabilistic matrix factorization. (b) Visualization of the first two factors in the PMF model estimated from the Netflix challenge data. Each movie $j$ is plotted at the location specified $\hat{\mathbf{v}}_j$. On the left we have low-brow humor and horror movies (*Half Baked*, *Freddy vs Jason*), and on the right we have more serious dramas (*Sophie's Choice*, *Moonstruck*). On the top we have critically acclaimed independent movies (*Punch-Drunk Love*, *I Heart Huckabees*), and on the bottom we have mainstream Hollywood blockbusters (*Armageddon*, *Runway Bride*). The *Wizard of Oz* is right in the middle of these axes. From Figure 3 of (Koren et al. 2009). Used with kind permission of Yehuda Koren.

This has been called **probabilistic matrix factorization** (PMF) (Salakhutdinov and Mnih 2008). See Figure 27.28(a) for the DGM. The intuition behind this method is that each user and each movie get embedded into the same low-dimensional continuous space (see Figure 27.28(b)). If a user is close to a movie in that space, they are likely to rate it highly. All of the best entries in the Netflix competition used this approach in one form or another.[8]

PMF is closely related to the SVD. In particular, if there is no missing data, then computing the MLE for the $\mathbf{u}_i$'s and the $\mathbf{v}_j$'s is equivalent to finding a rank $K$ approximation to $\mathbf{R}$. However, as soon as we have missing data, the problem becomes non-convex, as shown in

8. The winning entry was actually an ensemble of different methods, including PMF, nearest neighbor methods, etc.

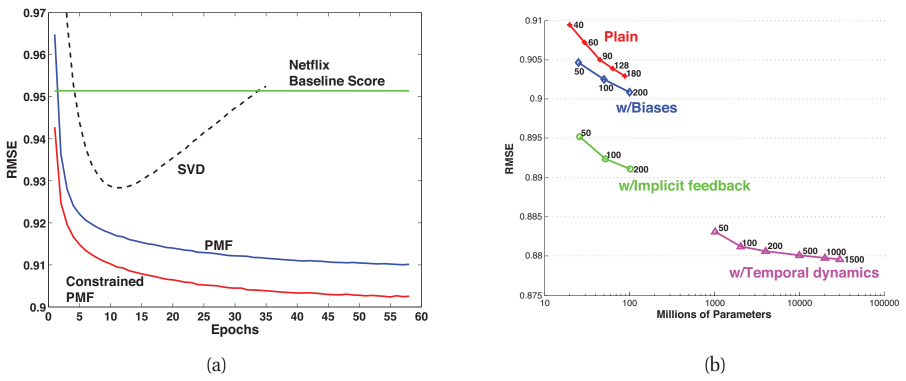


**Figure 27.29** (a) RMSE on the validation set for different PMF variants vs number of passes through the data. "SVD" is the unregularized version, $\lambda_U = \lambda_V = 0$. "PMF1" corresponds to $\lambda_U = 0.01$ and $\lambda_V = 0.001$, while "PMF2" corresponds to $\lambda_U = 0.001$ and $\lambda_V = 0.0001$. "PMFA1" corresponds to a version where the mean and diagonal covariance of the Gaussian prior were learned from data. From Figure 2 of (Salakhutdinov and Mnih 2008). Used with kind permission of Ruslan Salakhutdinov. (b) RMSE on the test set (quiz portion) vs number of parameters for several different models. "Plain" is the baseline PMF with suitably chosen $\lambda_U$, $\lambda_V$. "With biases" adds $f_i$ and $g_j$ offset terms. "With implicit feedback" "With temporal dynamics" allows the offset terms to change over time. The Netflix baseline system achieves an RMSE of 0.9514, and the grand prize's required accuracy is 0.8563 (which was obtained on 21 September 2009). Figure generated by `netflixResultsPlot`. From Figure 4 of (Koren et al. 2009). Used with kind permission of Yehuda Koren.

(Srebro and Jaakkola 2003), and standard SVD methods cannot be applied. (Recall that in the Netflix challenge, only about 1% of the matrix is observed.)

The most straightforward way to fit the PMF model is to minimize the overall NLL:

$$J(\mathbf{U}, \mathbf{V}) = -\log p(\mathbf{R}|\mathbf{U}, \mathbf{V}, \mathbf{O}) = -\log \left( \prod_{i=1}^{N} \prod_{j=1}^{M} \left[ \mathcal{N}(R_{ij}|\mathbf{u}_i^T \mathbf{v}_j, \sigma^2) \right]^{\mathbb{I}(O_{ij}=1)} \right) \tag{27.87}$$

where $O_{ij} = 1$ if user $i$ has seen movie $j$. Since this is non-convex, we can just find a locally optimal MLE. Since the Netflix data is so large (about 100 million observed entries), it is common to use stochastic gradient descent (Section 8.5.2) for this task. The gradient for $\mathbf{u}_i$ is given by

$$\frac{dJ}{d\mathbf{u}_i} = \frac{d}{d\mathbf{u}_i} \frac{1}{2} \sum_{ij} \mathbb{I}(O_{ij} = 1)(R_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2 = - \sum_{j:O_{ij}=1} e_{ij} \mathbf{v}_j \tag{27.88}$$

where $e_{ij} = R_{ij} - \mathbf{u}_i^T \mathbf{v}_j$ is the error term. By stochastically sampling a single movie $j$ that user $i$ has watched, the update takes the following simple form:

$$\mathbf{u}_i = \mathbf{u}_i + \eta e_{ij} \mathbf{v}_j \tag{27.89}$$

where $\eta$ is the learning rate. The update for $\mathbf{v}_j$ is similar.

Of course, just maximizing the likelihood results in overfitting, as shown in Figure 27.29(a). We can regularize this by imposing Gaussian priors:

$$p(\mathbf{U}, \mathbf{V}) = \prod_i \mathcal{N}(\mathbf{u}_i|\boldsymbol{\mu}_u, \boldsymbol{\Sigma}_u) \prod_j \mathcal{N}(\mathbf{v}_j|\boldsymbol{\mu}_v, \boldsymbol{\Sigma}_v) \tag{27.90}$$

If we use $\boldsymbol{\mu}_u = \boldsymbol{\mu}_v = \mathbf{0}$, $\boldsymbol{\Sigma}_u = \sigma_U^2 \mathbf{I}_K$, and $\boldsymbol{\Sigma}_v = \sigma_V^2 \mathbf{I}_K$, the new objective becomes

$$\begin{aligned} J(\mathbf{U}, \mathbf{V}) &= -\log p(\mathbf{R}, \mathbf{U}, \mathbf{V}|\mathbf{O}, \boldsymbol{\theta}) & (27.91) \\ &= \sum_i \sum_j \mathbb{I}(O_{ij} = 1)(R_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2 \\ &\quad +\lambda_U \sum_i ||\mathbf{u}_i||_2^2 + \lambda_V \sum_j ||\mathbf{v}_j||_2^2 + \text{const} & (27.92) \end{aligned}$$

where we have defined $\lambda_U = \sigma^2/\sigma_U^2$ and $\lambda_V = \sigma^2/\sigma_V^2$. By varying the regularizers, we can reduce the effect of overfitting, as shown in Figure 27.29(a). We can find MAP estimates using stochastic gradient descent. We can also compute approximate posteriors using variational Bayes (Ilin and Raiko 2010).

If we use diagonal covariances for the priors, we can penalize each latent dimension by a different amount. Also, if we use non-zero means for the priors, we can account for offset terms. Optimizing the prior parameters $(\boldsymbol{\mu}_u, \boldsymbol{\Sigma}_u, \boldsymbol{\mu}_v, \boldsymbol{\Sigma}_v)$ at the same time as the model parameters $(\mathbf{U}, \mathbf{V}, \sigma^2)$ is a way to create an adaptive prior. This avoids the need to search for the optimal values of $\lambda_U$ and $\lambda_V$, and gives even better results, as shown in Figure 27.29(a).

It turns out that much of the variation in the data can be explained by movie-specific or user-specific effects. For example, some movies are popular for all types of users. And some users give low scores for all types of movies. We can model this by allowing for user and movie specific offset or bias terms as follows:

$$p(R_{ij} = r|\mathbf{u}_i, \mathbf{v}_j, \boldsymbol{\theta}) = \mathcal{N}(r|\mathbf{u}_i^T \mathbf{v}_j + \mu + f_i + g_j, \sigma^2) \tag{27.93}$$

where $\mu$ is the overall mean, $f_i$ is the user bias, $g_j$ is the movie bias, and $\mathbf{u}_i^T \mathbf{v}_j$ is the interaction term. This is equivalent to applying PMF just to the residual matrix, and gives much better results, as shown in Figure 27.29(b). We can estimate the $f_i$, $g_j$ and $\mu$ terms using stochastic gradient descent, just as we estimated $\mathbf{U}$, $\mathbf{V}$ and $\boldsymbol{\theta}$.

We can also allow the bias terms to evolve over time, to reflect the changing preferences of users (Koren 2009b). This is important since in the Netflix competition, the test data was more recent than the training data. Figure 27.29(b) shows that allowing for temporal dynamics can help a lot.

Often we also have **side information** of various kinds. In the Netflix competition, entrants knew which movies the user had rated in the test set, even though they did not know the values of these ratings. That is, they knew the value of the (dense) $\mathbf{O}$ matrix even on the test set. If a user chooses to rate a movie, it is likely because they have seen it, which in turns means they thought they would like it. Thus the very act of rating reveals information. Conversely, if a user chooses not rate a movie, it suggests they knew they would not like it. So the data is not missing at random (see e.g., (Marlin and Zemel 2009)). Exploiting this can improve performance, as shown in Figure 27.29(b). In real problems, information on the test set is not available. However, we often know which movies the user has watched or declined to watch, even if they did not rate them (this is called **implicit feedback**), and this can be used as useful side information.

Another source of side information concerns the content of the movie, such as the movie genre, the list of the actors, or a synopsis of the plot. This can be denoted by $\mathbf{x}_v$, the features of the video. (In the case where we just have the id of the video, we can treat $\mathbf{x}_v$ as a $|\mathcal{V}|$-dimensional bit vector with just one bit turned on.) We may also know features about the user, which we can denote by $\mathbf{x}_u$. In some cases, we only know if the user clicked on the video or not, that is, we may not have a numerical rating. We can then modify the model as follows:

$$p(R(u,v)|\mathbf{x}_u, \mathbf{x}_v, \boldsymbol{\theta}) = \text{Ber}(R(u,v)|(\mathbf{U}\mathbf{x}_u)^T(\mathbf{V}\mathbf{x}_v)) \tag{27.94}$$

where $\mathbf{U}$ is a $|\mathcal{U}| \times K$ matrix, and $\mathbf{V}$ is a $|\mathcal{V}| \times K$ matrix (we can incorporate an offset term by appending a 1 to $\mathbf{x}_u$ and $\mathbf{x}_v$ in the usual way). A method for computing the approximate posterior $p(\mathbf{U}, \mathbf{V}|\mathcal{D})$ in an online fashion, using ADF and EP, was described in (Stern et al. 2009). This was implemented by Microsoft and has been deployed to predict click through rates on all the ads used by Bing.

Unfortunately, fitting this model just from positive binary data can result in an over prediction of links, since no negative examples are included. Better performance is obtained if one has access to the set of all videos shown to the user, of which at most one was picked; data of this form is known as an **impression log**. In this case, we can use a multinomial model instead of a binary model; in (Yang et al. 2011), this was shown to work much better than a binary model. To understand why, suppose some is presented with a choice of an action movie starring Arnold Schwarzenegger, an action movie starring Vin Diesel, and a comedy starring Hugh Grant. If the user picks Arnold Schwarzenegger, we learn not only that they like prefer action movies to comedies, but also that they prefer Schwarzenegger to Diesel. This is more informative than just knowing that they like Schwarzenegger and action movies.

## 27.7 Restricted Boltzmann machines (RBMs)

So far, all the models we have proposed in this chapter have been representable by directed graphical models. But some models are better represented using undirected graphs. For example, the **Boltzmann machine** (Ackley et al. 1985) is a pairwise MRF with hidden nodes $\mathbf{h}$ and visible nodes $\mathbf{v}$, as shown in Figure 27.30(a). The main problem with the Boltzmann machine is that exact inference is intractable, and even approximate inference, using e.g., Gibbs sampling, can be slow. However, suppose we restrict the architecture so that the nodes are arranged in layers, and so that there are no connections between nodes within the same layer (see Figure 27.30(b)). Then the model has the form

$$p(\mathbf{h}, \mathbf{v}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{r=1}^{R} \prod_{k=1}^{K} \psi_{rk}(v_r, h_k) \tag{27.95}$$

where $R$ is the number of visible (response) variables, $K$ is the number of hidden variables, and $\mathbf{v}$ plays the role of $\mathbf{y}$ earlier in this chapter. This model is known as a **restricted Boltzmann machine** (**RBM**) (Hinton 2002), or a **harmonium** (Smolensky 1986).

An RBM is a special case of a **product of experts** (PoE) (Hinton 1999), which is so-called because we are multiplying together a set of "experts" (here, potential functions on each edge)

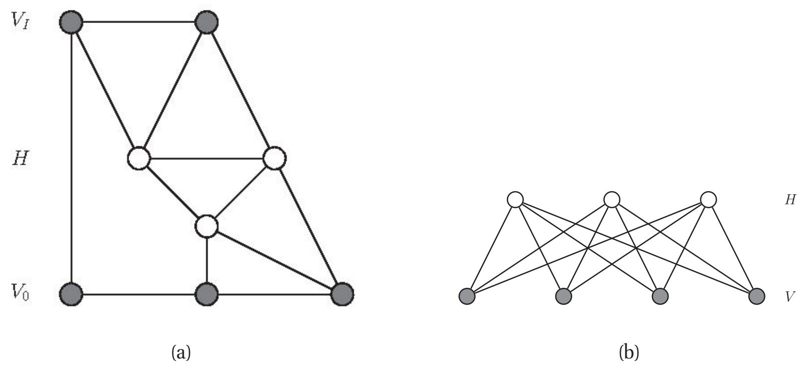


**Figure 27.30** (a) A general Boltzmann machine, with an arbitrary graph structure. The shaded (visible) nodes are partitioned into input and output, although the model is actually symmetric and defines a joint density on all the nodes. (b) A restricted Boltzmann machine with a bipartite structure. Note the lack of intra-layer connections.

and then normalizing, whereas in a mixture of experts, we take a convex combination of normalized distributions. The intuitive reason why PoE models might work better than a mixture is that each expert can enforce a constraint (if the expert has a value which is $\gg 1$ or $\ll 1$) or a "don't care" condition (if the expert has value 1). By multiplying these experts together in different ways we can create "sharp" distributions which predict data which satisfies the specified constraints (Hinton and Teh 2001). For example, consider a distributed model of text. A given document might have the topics "government", "mafia" and "playboy". If we "multiply" the predictions of each topic together, the model may give very high probability to the word "Berlusconi"[9] (Salakhutdinov and Hinton 2010). By contrast, adding together experts can only make the distribution broader (see Figure 14.17).

Typically the hidden nodes in an RBM are binary, so $\mathbf{h}$ specifies which constraints are active. It is worth comparing this with the directed models we have discussed. In a mixture model, we have one hidden variable $q \in \{1, \ldots, K\}$. We can represent this using a set of $K$ bits, with the restriction that exactly one bit is on at a time. This is called a **localist encoding**, since only one hidden unit is used to generate the response vector. This is analogous to the hypothetical notion of **grandmother cells** in the brain, that are able to recognize only one kind of object. By contrast, an RBM uses a **distributed encoding**, where many units are involved in generating each output. Models that used vector-valued hidden variables, such as $\boldsymbol{\pi} \in S_K$, as in mPCA/LDA, or $\mathbf{z} \in \mathbb{R}^K$, as in ePCA also use distributed encodings.

The main difference between an RBM and directed two-layer models is that the hidden variables are conditionally independent given the visible variables, so the posterior factorizes:

$$p(\mathbf{h}|\mathbf{v}, \boldsymbol{\theta}) = \prod_k p(h_k|\mathbf{v}, \boldsymbol{\theta}) \tag{27.96}$$

This makes inference much simpler than in a directed model, since we can estimate each $h_k$

9. Silvio Berlusconi is the current (2011) prime minister of Italy.

| Visible | Hidden | Name | Reference |
|---|---|---|---|
| Binary | Binary | Binary RBM | (Hinton 2002) |
| Gaussian | Binary | Gaussian RBM | (Welling and Sutton 2005) |
| Categorical | Binary | Categorical RBM | (Salakhutdinov et al. 2007) |
| Multiple categorical | Binary | Replicated softmax/ undirected LDA | (Salakhutdinov and Hinton 2010) |
| Gaussian | Gaussian | Undirected PCA | (Marks and Movellan 2001) |
| Binary | Gaussian | Undirected binary PCA | (Welling and Sutton 2005) |

**Table 27.2** Summary of different kinds of RBM.

independently and in parallel, as in a feedforward neural network. The disadvantage is that training undirected models is much harder, as we discuss below.

## 27.7.1 Varieties of RBMs

In this section, we describe various forms of RBMs, by defining different pairwise potential functions. See Table 27.2 for a summary. All of these are special cases of the **exponential family harmonium** (Welling et al. 2004).

### 27.7.1.1 Binary RBMs

The most common form of RBM has binary hidden nodes and binary visible nodes. The joint distribution then has the following form:

$$p(\mathbf{v},\mathbf{h}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\exp(-E(\mathbf{v},\mathbf{h};\boldsymbol{\theta})) \tag{27.97}$$

$$E(\mathbf{v},\mathbf{h};\boldsymbol{\theta}) \triangleq -\sum_{r=1}^{R}\sum_{k=1}^{K} v_r h_k W_{rk} - \sum_{r=1}^{R} v_r b_r - \sum_{k=1}^{K} h_k c_k \tag{27.98}$$

$$= -(\mathbf{v}^T\mathbf{W}\mathbf{h} + \mathbf{v}^T\mathbf{b} + \mathbf{h}^T\mathbf{c}) \tag{27.99}$$

$$Z(\boldsymbol{\theta}) = \sum_{\mathbf{v}}\sum_{\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h};\boldsymbol{\theta})) \tag{27.100}$$

where $E$ is the energy function, $\mathbf{W}$ is a $R \times K$ weight matrix, $\mathbf{b}$ are the visible bias terms, $\mathbf{c}$ are the hidden bias terms, and $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b}, \mathbf{c})$ are all the parameters. For notational simplicity, we will absorb the bias terms into the weight matrix by clamping dummy units $v_0 = 1$ and $h_0 = 1$ and setting $\mathbf{w}_{0,:} = \mathbf{c}$ and $\mathbf{w}_{:,0} = \mathbf{b}$. Note that naively computing $Z(\boldsymbol{\theta})$ takes $O(2^R 2^K)$ time but we can reduce this to $O(\min\{R2^K, K2^R\})$ time (Exercise 27.1).

When using a binary RBM, the posterior can be computed as follows:

$$p(\mathbf{h}|\mathbf{v},\boldsymbol{\theta}) = \prod_{k=1}^{K} p(h_k|\mathbf{v},\boldsymbol{\theta}) = \prod_k \text{Ber}(h_k|\text{sigm}(\mathbf{w}_{:,k}^T\mathbf{v})) \tag{27.101}$$

By symmetry, one can show that we can generate data given the hidden variables as follows:

$$p(\mathbf{v}|\mathbf{h},\boldsymbol{\theta}) = \prod_r p(v_r|\mathbf{h},\boldsymbol{\theta}) = \prod_r \text{Ber}(v_r|\text{sigm}(\mathbf{w}_{r,:}^T\mathbf{h})) \tag{27.102}$$

We can write this in matrix-vetor notation as follows:

$$\mathbb{E}[\mathbf{h}|\mathbf{v}\boldsymbol{\theta}] = \text{sigm}(\mathbf{W}^T\mathbf{v}) \quad (27.103)$$

$$\mathbb{E}[\mathbf{v}|\mathbf{h}, \boldsymbol{\theta}] = \text{sigm}(\mathbf{W}\mathbf{h}) \quad (27.104)$$

The weights in $\mathbf{W}$ are called the **generative weights**, since they are used to generate the observations, and the weights in $\mathbf{W}^T$ are called the **recognition weights**, since they are used to recognize the input.

From Equation 27.101, we see that we activate hidden node $k$ in proportion to how much the input vector $\mathbf{v}$ "looks like" the weight vector $\mathbf{w}_{:,k}$ (up to scaling factors). Thus each hidden node captures certain features of the input, as encoded in its weight vector, similar to a feedforward neural network.

#### 27.7.1.2 Categorical RBM

We can extend the binary RBM to categorical visible variables by using a 1-of-$C$ encoding, where $C$ is the number of states for each $v_{ir}$. We define a new energy function as follows (Salakhutdinov et al. 2007; Salakhutdinov and Hinton 2010):

$$E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) \triangleq -\sum_{r=1}^{R}\sum_{k=1}^{K}\sum_{c=1}^{C} v_r^c h_k W_{rk}^c - \sum_{r=1}^{R}\sum_{c=1}^{C} v_r^c b_r^c - \sum_{k=1}^{K} h_k c_k \quad (27.105)$$

The full conditionals are given by

$$p(v_r|\mathbf{h}, \boldsymbol{\theta}) = \text{Cat}(\mathcal{S}(\{b_r^c + \sum_k h_k W_{rk}^c\}_{c=1}^C)) \quad (27.106)$$

$$p(h_k = 1|\mathbf{c}, \boldsymbol{\theta}) = \text{sigm}(c_k + \sum_r \sum_c v_r^c W_{rk}^c) \quad (27.107)$$

#### 27.7.1.3 Gaussian RBM

We can generalize the model to handle real-valued data. In particular, a **Gaussian RBM** has the following energy function:

$$E(\mathbf{v}, \mathbf{h}|\boldsymbol{\theta}) = -\sum_{r=1}^{R}\sum_{k=1}^{K} W_{rk} h_k v_r - \frac{1}{2}\sum_{r=1}^{R}(v_r - b_r)^2 - \sum_{k=1}^{K} a_k h_k \quad (27.108)$$

The parameters of the model are $\boldsymbol{\theta} = (w_{rk}, a_k, b_r)$. (We have assumed the data is standardized, so we fix the variance to $\sigma^2 = 1$.) Compare this to a Gaussian in information form:

$$\mathcal{N}_c(\mathbf{v}|\boldsymbol{\eta}, \boldsymbol{\Lambda}) \propto \exp(\boldsymbol{\eta}^T\mathbf{v} - \frac{1}{2}\mathbf{v}^T\boldsymbol{\Lambda}\mathbf{v}) \quad (27.109)$$

where $\boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}$. We see that we have set $\boldsymbol{\Lambda} = \mathbf{I}$, and $\boldsymbol{\eta} = \sum_k h_k \mathbf{w}_{:,k}$. Thus the mean is given by $\boldsymbol{\mu} = \boldsymbol{\Lambda}^{-1}\boldsymbol{\eta} = \sum_k h_k \mathbf{w}_{:,k}$. The full conditionals, which are needed for inference and learning, are given by

$$p(v_r|\mathbf{h}, \boldsymbol{\theta}) = \mathcal{N}(v_r|b_r + \sum_k w_{rk} h_k, 1) \tag{27.110}$$

$$p(h_k = 1|\mathbf{v}, \boldsymbol{\theta}) = \text{sigm}\left(c_k + \sum_r w_{rk} v_r\right) \tag{27.111}$$

We see that each visible unit has a Gaussian distribution whose mean is a function of the hidden bit vector. More powerful models, which make the (co)variance depend on the hidden state, can also be developed (Ranzato and Hinton 2010).

#### 27.7.1.4 RBMs with Gaussian hidden units

If we use Gaussian latent variables and Gaussian visible variables, we get an undirected version of factor analysis. However, it turns out that it is identical to the standard directed version (Marks and Movellan 2001).

If we use Gaussian latent variables and categorical observed variables, we get an undirected version of categorical PCA (Section 27.2.2). In (Salakhutdinov et al. 2007), this was applied to the Netflix collaborative filtering problem, but was found to be significantly inferior to using binary latent variables, which have more expressive power.

### 27.7.2 Learning RBMs

In this section, we discuss some ways to compute ML parameter estimates of RBMs, using gradient-based optimizers. It is common to use stochastic gradient descent, since RBMs often have many parameters and therefore need to be trained on very large datasets. In addition, it is standard to use $\ell_2$ regularization, a technique that is often called weight decay in this context. This requires a very small change to the objective and gradient, as discussed in Section 8.3.6.

#### 27.7.2.1 Deriving the gradient using $p(\mathbf{h}, \mathbf{v}|\boldsymbol{\theta})$

To compute the gradient, we can modify the equations from Section 19.5.2, which show how to fit a generic latent variable maxent model. In the context of the Boltzmann machine, we have one feature per edge, so the gradient is given by

$$\frac{\partial \ell}{\partial w_{rk}} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}\left[v_r h_k|\mathbf{v}_i, \boldsymbol{\theta}\right] - \mathbb{E}\left[v_r h_k|\boldsymbol{\theta}\right] \tag{27.112}$$

We can write this in matrix-vector form as follows:

$$\nabla_{\mathbf{w}} \ell = \mathbb{E}_{p_{\text{emp}}(\cdot|\boldsymbol{\theta})}\left[\mathbf{v}\mathbf{h}^T\right] - \mathbb{E}_{p(\cdot|\boldsymbol{\theta})}\left[\mathbf{v}\mathbf{h}^T\right] \tag{27.113}$$

where $p_{\text{emp}}(\mathbf{v}, \mathbf{h}|\boldsymbol{\theta}) \triangleq p(\mathbf{h}|\mathbf{v}, \boldsymbol{\theta}) p_{\text{emp}}(\mathbf{v})$, and $p_{\text{emp}}(\mathbf{v}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\mathbf{v}_i}(\mathbf{v})$ is the empirical distribution. (We can derive a similar expression for the bias terms by setting $v_r = 1$ or $h_k = 1$.)

The first term on the gradient, when $\mathbf{v}$ is fixed to a data case, is sometimes called the **clamped phase**, and the second term, when $\mathbf{v}$ is free, is sometimes called the **unclamped**

**phase**. When the model expectations match the empirical expectations, the two terms cancel out, the gradient becomes zero and learning stops. This algorithm was first proposed in (Ackley et al. 1985). The main problem is efficiently computing the expectations. We discuss some ways to do this below.

#### 27.7.2.2 Deriving the gradient using $p(\mathbf{v}|\boldsymbol{\theta})$

We now present an alternative way to derive Equation 27.112, which also applies to other energy based models. First we marginalize out the hidden variables and write the RBM in the form $p(\mathbf{v}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp(-F(\mathbf{v};\boldsymbol{\theta}))$, where $F(\mathbf{v};\boldsymbol{\theta})$ is the **free energy**:

$$\begin{aligned} F(\mathbf{v}) & \triangleq \sum_{\mathbf{h}} E(\mathbf{v},\mathbf{h}) = \sum_{\mathbf{h}} \exp\left(\sum_{r=1}^{R}\sum_{k=1}^{K} v_r h_k W_{rk}\right) && (27.114) \\ &= \sum_{\mathbf{h}} \prod_{k=1}^{K} \exp\left(\sum_{r=1}^{R} v_r h_k W_{rk}\right) && (27.115) \\ &= \prod_{k=1}^{K} \sum_{h_r \in \{0,1\}} \exp\left(\sum_{r=1}^{R} v_r h_r W_{rk}\right) && (27.116) \\ &= \prod_{k=1}^{K} \left(1 + \exp(\sum_{r=1}^{R} v_r W_{rk})\right) && (27.117) \end{aligned}$$

Next we write the (scaled) log-likelihood in the following form:

$$\ell(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \log p(\mathbf{v}_i|\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N} F(\mathbf{v}_i|\boldsymbol{\theta}) - \log Z(\boldsymbol{\theta}) \tag{27.118}$$

Using the fact that $Z(\boldsymbol{\theta}) = \sum_{\mathbf{v}} \exp(-F(\mathbf{v};\boldsymbol{\theta}))$ we have

$$\begin{aligned} \nabla \ell(\boldsymbol{\theta}) &= -\frac{1}{N}\sum_{i=1}^{N} \nabla F(\mathbf{v}_i) - \frac{\nabla Z}{Z} && (27.119) \\ &= -\frac{1}{N}\sum_{i=1}^{N} \nabla F(\mathbf{v}_i) + \sum_{\mathbf{v}} \nabla F(\mathbf{v}) \frac{\exp(-F(\mathbf{v}))}{Z} && (27.120) \\ &= -\frac{1}{N}\sum_{i=1}^{N} \nabla F(\mathbf{v}_i) + \mathbb{E}\left[\nabla F(\mathbf{v})\right] && (27.121) \end{aligned}$$

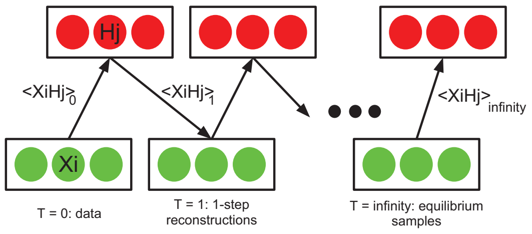
Plugging in the free energy (Equation 27.117), one can show that

$$\frac{\partial}{\partial w_{rk}} F(\mathbf{v}) = -v_r \mathbb{E}\left[h_k|\mathbf{v},\boldsymbol{\theta}\right] = -\mathbb{E}\left[v_r h_k|\mathbf{v},\boldsymbol{\theta}\right] \tag{27.122}$$

Hence

$$\frac{\partial}{\partial w_{rk}} \ell(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}\left[v_r h_k|\mathbf{v},\boldsymbol{\theta}\right] - \mathbb{E}\left[v_r h_k|\boldsymbol{\theta}\right] \tag{27.123}$$

which matches Equation 27.112.



Hj
Xi
<XiHj>0
<XiHj>1
<XiHj>infinity
T = 0: data
T = 1: 1-step reconstructions
T = infinity: equilibrium samples

**Figure 27.31** Illustration of Gibbs sampling in an RBM. The visible nodes are initialized at a datavector, then we sample a hidden vector, then another visible vector, etc. Eventually (at "infinity") we will be producing samples from the joint distribution $p(\mathbf{v}, \mathbf{h}|\boldsymbol{\theta})$.

#### 27.7.2.3 Approximating the expectations

We can approximate the expectations needed to evaluate the gradient by performing block Gibbs sampling, using Equations 27.101 and 27.102. In more detail, we can sample from the joint distribution $p(\mathbf{v}, \mathbf{h}|\boldsymbol{\theta})$ as follows: initialize the chain at $vv^1$ (e.g. by setting $\mathbf{v}^1 = \mathbf{v}_i$ for some data vector), and then sample from $\mathbf{h}^1 \sim p(\mathbf{h}|\mathbf{v}^1)$, then from $\mathbf{v}^2 \sim p(\mathbf{v}|\mathbf{h}^1)$, then from $\mathbf{h}^2 \sim p(\mathbf{h}|\mathbf{v}^2)$, etc. See Figure 27.31 for an illustration. Note, however, that we have to wait until the Markov chain reaches equilibrium (i.e., until it has "burned in") before we can interpret the samples as coming from the joint distribution of interest, and this might take a long time.

A faster alternative is to use mean field, where we make the approximation $\mathbb{E}[v_r h_k] \approx \mathbb{E}[v_r]\,\mathbb{E}[h_k]$. However, since $p(\mathbf{v}, \mathbf{h})$ is typically multimodal, this is usually a very poor approximation, since it will average over different modes (see Section 21.2.2). Furthermore, there is a more subtle reason not to use mean field: since the gradient has the form $\mathbb{E}[v_r h_k|\mathbf{v}] - \mathbb{E}[v_r h_k]$, we see that the negative sign in front means that the method will try to make the variational bound as loose as possible (Salakhutdinov and Hinton 2009). This explains why earlier attempts to use mean field to learn Boltzmann machines (e.g., (Kappen and Rodriguez 1998)) did not work.

#### 27.7.2.4 Contrastive divergence

The problem with using Gibbs sampling to compute the gradient is that it is slow. We now present a faster method known as **contrastive divergence** or **CD** (Hinton 2002). CD was originally derived by approximating an objective function defined as the difference of two KL divergences, rather than trying to maximize the likelihood itself. However, from an algorithmic point of view, it can be thought of as similar to stochastic gradient descent, except it approximates the "unclamped" expectations with "brief" Gibbs sampling where we initialize each Markov chain at the data vectors. That is, we approximate the gradient for one datavector as follows:

$$\nabla_{\mathbf{w}}\, \ell \approx \mathbb{E}\left[\mathbf{v}\mathbf{h}^T|\mathbf{v}_i\right] - \mathbb{E}_q\left[\mathbf{v}\mathbf{h}^T\right] \tag{27.124}$$

where $q$ corresponds to the distribution generated by $K$ up-down Gibbs sweeps, started at $\mathbf{v}_i$, as in Figure 27.31. This is known as CD-$K$. In more detail, the procedure (for $K = 1$) is as follows:

$$\mathbf{h}_i \sim p(\mathbf{h}|\mathbf{v}_i, \boldsymbol{\theta}) \quad (27.125)$$
$$\mathbf{v}'_i \sim p(\mathbf{v}|\mathbf{h}_i, \boldsymbol{\theta}) \quad (27.126)$$
$$\mathbf{h}'_i \sim p(\mathbf{h}|\mathbf{v}'_i, \boldsymbol{\theta}) \quad (27.127)$$

We then make the approximation

$$\mathbb{E}_q\left[\mathbf{v}\mathbf{h}^T\right] \approx \mathbf{v}_i(\mathbf{h}'_i)^T \quad (27.128)$$

Such samples are sometimes called **fantasy data**. We can think of $\mathbf{v}'_i$ as the model's best attempt at reconstructing $\mathbf{v}_i$ after being coded and then decoded by the model. This is similar to the way we train auto-encoders, which are models which try to "squeeze" the data through a restricted parametric "bottleneck" (see Section 28.3.2).

In practice, it is common to use $\mathbb{E}[\mathbf{h}|\mathbf{v}'_i]$ instead of a sampled value $\mathbf{h}'_i$ in the final upwards pass, since this reduces the variance. However, it is not valid to use $\mathbb{E}[\mathbf{h}|\mathbf{v}_i]$ instead of sampling $\mathbf{h}_i \sim p(\mathbf{h}|\mathbf{v}_i)$ in the earlier upwards passes, because then each hidden unit would be able to pass more than 1 bit of information, so it would not act as much of a bottleneck.

The whole procedure is summarized in Algorithm 3. (Note that we follow the positive gradient since we are maximizing likelihood.) Various tricks can be used to speed this algorithm up, such as using a momentum term (Section 8.3.2), using mini-batches, averaging the updates, etc. Such details can be found in (Hinton 2010; Swersky et al. 2010).

**Algorithm 27.3:** CD-1 training for an RBM with binary hidden and visible units

```
1  Initialize weights $\mathbf{W} \in \mathbb{R}^{R \times K}$ randomly;
2  $t := 0$;
3  for each epoch do
4      $t := t + 1$ ;
5      for each minibatch of size $B$ do
6          Set minibatch gradient to zero, $\mathbf{g} := \mathbf{0}$ ;
7          for each case $\mathbf{v}_i$ in the minibatch do
8              Compute $\boldsymbol{\mu}_i = \mathbb{E}[\mathbf{h}|\mathbf{v}_i, \mathbf{W}]$;
9              Sample $\mathbf{h}_i \sim p(\mathbf{h}|\mathbf{v}_i, \mathbf{W})$;
10             Sample $\mathbf{v}'_i \sim p(\mathbf{v}|\mathbf{h}_i, \mathbf{W})$;
11             Compute $\boldsymbol{\mu}'_i = \mathbb{E}[\mathbf{h}|\mathbf{v}'_i, \mathbf{W}]$;
12             Compute gradient $\nabla_{\mathbf{W}} = (\mathbf{v}_i)(\boldsymbol{\mu}_i)^T - (\mathbf{v}'_i)(\boldsymbol{\mu}'_i)^T$ ;
13             Accumulate $\mathbf{g} := \mathbf{g} + \nabla_{\mathbf{W}}$;
14         Update parameters $\mathbf{W} := \mathbf{W} + (\alpha_t/B)\mathbf{g}$
```

### 27.7.2.5 Persistent CD

In Section 19.5.5, we presented a technique called stochastic maximum likelihood (SML) for fitting maxent models. This avoids the need to run MCMC to convergence at each iteration, by exploiting the fact that the parameters are changing slowly, so the Markov chains will not be pushed too far from equilibrium after each update (Younes 1989). In other words, there are two dynamical processes running at different time scales: the states change quickly, and the parameters change slowly. This algorithm was independently rediscovered in (Tieleman 2008), who called it **persistent CD**. See Algorithm 3 for the pseudocode.

PCD often works better than CD (see e.g., (Tieleman 2008; Marlin et al. 2010; Swersky et al. 2010)), although CD can be faster in the early stages of learning.

**Algorithm 27.4:** Persistent CD for training an RBM with binary hidden and visible units

```
1  Initialize weights $\mathbf{W} \in \mathbb{R}^{D \times L}$ randomly;
2  Initialize chains $(\mathbf{v}_s, \mathbf{h}_s)_{s=1}^S$ randomly ;
3  for $t = 1, 2, \ldots$ do
4      // Mean field updates ;
5      for each case $i = 1 : N$ do
6          $\mu_{ik} = \text{sigm}(\mathbf{v}_i^T \mathbf{w}_{:,k})$
7      // MCMC updates ;
8      for each sample $s = 1 : S$ do
9          Generate $(\mathbf{v}_s, \mathbf{h}_s)$ by brief Gibbs sampling from old $(\mathbf{v}_s, \mathbf{h}_s)$
10     // Parameter updates ;
11     $\mathbf{g} = \frac{1}{N} \sum_{i=1}^N \mathbf{v}_i (\boldsymbol{\mu}_i)^T - \frac{1}{S} \sum_{s=1}^S \mathbf{v}_s (\mathbf{h}_s)^T$ ;
12     $\mathbf{W} := \mathbf{W} + \alpha_t \mathbf{g}$;
13     Decrease $\alpha_t$
```

### 27.7.3 Applications of RBMs

The main application of RBMs is as a building block for deep generative models, which we discuss in Section 28.2. But they can also be used as substitutes for directed two-layer models. They are particularly useful in cases where inference of the hidden states at test time must be fast. We give some examples below.

#### 27.7.3.1 Language modeling and document retrieval

We can use a categorical RBM to define a generative model for bag-of-words, as an alternative to LDA. One subtlety is that the partition function in an undirected models depends on how big the graph is, and therefore on how long the document is. A solution to this was proposed in (Salakhutdinov and Hinton 2010): use a categorical RBM with tied weights, but multiply the hidden activation bias terms $c_k$ by the document length $L$ to compensate form the fact that the observed word-count vector $\mathbf{v}$ is larger in magnitude:

$$E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) \triangleq -\sum_{k=1}^K \sum_{c=1}^C v^c h_k W_k^c - \sum_{c=1}^C v^c b_r^c - L \sum_{k=1}^K h_k c_k \tag{27.129}$$