

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

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 $y_{il} \in \{1, \dots, V\}$ 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 $n_{iv} \in \{0, 1, \dots, L_i\}$ to denote the number of times word v occurs in document i , for $v = 1: V$. Note that the $N \times V$ count matrix 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_{ir} , for $r = 1: R$ (the number of responses) and $i = 1: L_r$. If $L_r = 1$, it means we have a single token (a bag of length 1); in this case, we just write $y_{ir} \in \{1, \dots, V_r\}$ for brevity. If every channel is just a single token, we write the fixed-size response vector as $y_{i1:R}$; in this case, the $N \times R$ design matrix 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. Our goal is to build joint probability models of $p(y_i)$ or $p(n_i)$ 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 R , reserving the symbol Y for any categorical data (e.g., text) associated with the nodes.

已思考 17s

27 离散数据的潜在变量模型

27.1 引言 (原文逐句全译)

在本章中，我们关心的是**离散数据**的潜在变量模型，例如比特向量、类别变量序列、计数向量、图结构、关系型数据，等等。这些模型可用于分析投票记录、文本与文档集合、低强度图像、电影评分等。不过，我们将主要聚焦于**文本分析**，术语也会据此进行表述。

由于要处理的数据类型很多，我们需要**精确的记号**来保持清晰。当我们对**可变长度的类别变量序列**（即符号或 token），例如文档中的词语，进行建模时，我们令 $y_{il} \in \{1, \dots, V\}$ 表示第 i 篇文档中第 l 个词的身份 (ID)，其中 V 是**词汇表**中可能词的数量。我们假设 $l = 1:L_i$ ，其中 L_i 是第 i 篇文档的 (已知) 长度，并且 $i = 1:N$ ，其中 N 是文档的数量。

我们经常**忽略词序**，从而得到**词袋 (bag of words)**。这可以被化为一个**固定长度的计数向量 (直方图)**。我们用 $n_{iv} \in \{0, 1, \dots, L_i\}$ 表示词 v 在第 i 篇文档中出现的次数，其中 $v = 1:V$ 。注意到， $N \times V$ 的**计数矩阵 N** 通常**维度很大但很稀疏**，因为我们往往有很多文档，但在任一给定文档中，大多数词都**不会出现**。

在某些情况下，我们可能拥有**多个不同的词袋**，例如**文本词袋与视觉词袋**。这些对应于不同的“**通道**”或**特征类型**。我们用 y_{irl} 来表示它们，其中 $r = 1:R$ (响应/通道的数量)， $l = 1:L_{ir}$ 。如果 $L_{ir} = 1$ ，这表示我们只有**单个 token (长度为 1 的词袋)**；在这种情况下，为了简洁，我们就写 $y_{ir} \in \{1, \dots, V_r\}$ 。如果**每个通道都只有一个 token**，我们把**固定大小的响应向量**写作 $y_{i,1:R}$ ；在这种情况下， $N \times R$ 的**设计矩阵 Y** 将不会是稀疏的。举例来说，在**社会科学调查**中， y_{ir} 可以是第 i 个人对第 r 个**多项选择题**的作答。

我们的目标是构建 $p(y_i)$ 或 $p(n_i)$ 的**联合概率模型**，并用**潜在变量来捕捉相关性**。随后我们将尝试**解释潜在变量**，它们为数据提供一种**压缩表示**。我们在第 27.2 节对若干方法做一个概览，之后再在后续各节中给出更多细节。

在本章的末尾，我们将考虑对图与关系进行建模，这些也可以表示为稀疏的离散矩阵。例如，我们可能想要建模“哪些论文会引用哪些其他论文”的图。我们用 **R** 表示这些关系，并保留符号 **Y** 表示与节点相关联的任何类别数据（例如文本）。

通俗详解（配合例子讲清楚）

1) 什么是“离散数据”？

离散数据的取值是可数的“格子点”，比如：

- 文本里的词 ID（第 1 个词、第 2 个词...到第 V 个词）；
- 问卷题目的选项编号（A/B/C/D）；
- 计数（某个词在文章里出现了几次）；
- 图里有没有一条边（0/1 表示是否相连）。

这些都不是连续的小数，而是一个个类别或个数。

2) 为什么要用“潜在变量”？

- 观测到的离散数据之间经常存在关联（例如，提到“篮球”的文章更可能同时提到“NBA”、“球员”）。
- 直接在高维离散空间里建模很难，因此我们假设数据背后有少量看不见的“因子/主题/群组”在驱动这些关联——这就是潜在变量（latent variables）。
- 学到这些潜在变量后，我们既可以解释（比如主题词是什么），也可以压缩表示（用几个潜在因子刻画一篇文章，而不是成千上万个词）。

3) 两种常见表示：序列 vs. 词袋

- 序列表示：保留词序，用 $y_{il} \in \{1, \dots, V\}$ 表示第 i 篇文档第 l 个词的ID；长度是 L_i 。
- 词袋表示：不看词序，仅统计每个词的出现次数 n_{iv} 。得到的就是一个直方图向量（维度 V ），对所有文档堆起来就是一个 $N \times V$ 的计数矩阵。
 - 这个矩阵很大（文档多、词也多），但很稀疏（多数格子是 0），因为每篇文档只用到极少数词。
 - 小例子：假设 3 篇短文 ($N = 3$)，词表 10000 个词 ($V = 10,000$)。每篇短文用到几十个词，那么 $3 \times 10,000$ 的矩阵里绝大多数位置都是 0。

注：原文把文档数量记作 N ，也把计数矩阵写成粗体 **N**。两个都是“N”，容易混淆。在很多书里会把计数矩阵记作 **X** 或 **C**。

4) 多通道数据（不止一种“词袋”）

- 现实里，有时同一个对象会有多种类别型特征：

- 一张图片既有**文本标签**（“猫、沙发”），又有**视觉词**（把局部图像片段聚类成“视觉单词”）；
- 一份调查问卷里有很多**选择题**。
- 这时用 y_{irl} 表示：第 i 个样本、第 r 个通道中的第 l 个 token。
 - 若某通道只有 1 个 token ($L_{ir} = 1$)，直接写成 $y_{ir} \in \{1, \dots, V_r\}$ 。
 - 若**每个通道都只有 1 个 token**，则第 i 个样本可写成**固定长度向量** $y_{i,1:R}$ （长度 R ），把所有样本叠成 $N \times R$ 的**设计矩阵** \mathbf{Y} 。
 - 由于每个位置都有一个具体的选项（不是计数）， \mathbf{Y} 通常**不稀疏**。
- 问卷示例： y_{ir} 就是**第 i 个人对第 r 道选择题选了哪个选项**（选项被编号 1 到 V_r ）。

5) 我们到底在建什么模型？

- 针对第 i 个样本（比如第 i 篇文档），可以建
 - **序列视角**： $p(y_i)$ （整篇文档所有词的联合分布）；
 - **词袋视角**： $p(n_i)$ （整篇文档计数向量的分布）。
- 在这两个分布里加入**潜在变量**（如“主题”或“群组”），让模型**解释**为什么某些词或某些通道的取值**会一起出现**（相关性）。
- 学到的潜在变量既能**降低维度**（把文档压缩成少量主题权重），又能**有意义地解释**（每个主题大致对应一个话题领域）。

6) 图与关系数据（本章后面会讲）

- 许多现实问题是**谁与谁有关**：论文**引用**关系、社交网络的**关注**关系、商品与用户的**交互**等。
- 这类数据自然形成一个**关系矩阵**（行=源节点，列=目标节点，条目是 0/1 或计数），同样**稀疏**（大多数对之间没有边）。
- 本章使用 \mathbf{R} 表示这种**关系**；而 \mathbf{Y} 保留给**节点的类别型属性**（如节点的文本描述）。
- 小例子：论文网中，若“论文 A 引用论文 B”，就在 \mathbf{R} 的 (A, B) 位置记 1，否则为 0。

原文“which papers mycite which other papers”里的“mycite”**很可能是排版错误**，应为“**might cite**”（“可能引用”）。

一点点直觉与微型例子

- **直觉：主题模型**（仅作直观说明）
 - 假设每篇文章由少量**主题**混合而成（比如“体育”“科技”“财经”），每个主题各自偏好一组词（在该主题下出现概率更高）。

- “体育”主题提升词“篮球、比赛、球员”的出现概率；“财经”主题提升“市场、利率、股票”的概率。
- 不同文章拥有不同的**主题权重**，这就是潜在表示（也可看作低维向量）。
- 通过这些潜在变量，模型自然能解释“哪些词容易一起出现”。
- 微型词袋例子**
 - 词表：{篮球, 比赛, 芯片, 市场}，即 $V = 4$ 。
 - 文档 1 (体育)：“篮球 比赛 篮球” → 计数 [2, 1, 0, 0]。
 - 文档 2 (财经)：“市场 芯片 市场” → 计数 [0, 0, 1, 2]。
 - 计数矩阵 ($N = 2$)：

$$\mathbf{N} = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

- 一个含“体育”“财经”两个潜在主题的模型就能**压缩**：每篇文档≈[体育 1.0, 财经 0.0] 或 [体育 0.0, 财经 1.0]，并**解释**为什么“篮球、比赛”会一起出现。

要点小结

- 对象**：离散数据（类别、计数、图/关系），本章侧重**文本**。
- 记号**：
 - 序列： $y_{il} \in \{1, \dots, V\}$, $l = 1:L_i$, $i = 1:N$ 。
 - 词袋：计数 n_{iv} ，形成**大而稀疏**的 $N \times V$ 计数矩阵 \mathbf{N} 。
 - 多通道： y_{irl} ；若每通道 1 个 token，则拼成 $N \times R$ 的**不稀疏**设计矩阵 \mathbf{Y} 。
- 目标**：构建 $p(y_i)$ 或 $p(n_i)$ 的联合概率模型，用**潜在变量来捕捉相关性并压缩表示数据**。
- 扩展到关系数据**：用 \mathbf{R} 表示**图/关系矩阵**（稀疏）， \mathbf{Y} 表示**节点的类别属性**。
- 小提示**：原文“mycite”应是“might cite”的笔误；同时 \mathbf{N} 既作文档数符号又作计数矩阵名，实际使用时注意区分。

如果你接下来想看具体模型（比如主题模型、混合模型、矩阵分解、图模型）怎么写与怎么训练，我可以继续把第 27.2 节的概览也用同样方式翻译+讲解。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

27.2 Distributed state LVMs for discrete data

In this section, we summarize a variety of possible approaches for constructing models of the form $p(y_i, 1:L_i)$, for bags of tokens; $p(y_1:R)$, for vectors of tokens; and $p(n_i)$, 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, $q_i \in \{1, \dots, K\}$, with every document, where K is the number of clusters. We will use a discrete prior, $q_i \sim \text{Cat}(\pi)$. 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 b_k is the k 'th topic's word distribution. That is, the likelihood has the form

$$p(y_i, 1:L_i | q_i = k) = \prod_{l=1}^{L_i} \text{Cat}(y_{il}|b_k) \quad (27.1)$$

The induced distribution on the visible data is given by
 $p(y_i, 1:L_i) = \sum_{k=1}^K \pi_k \prod_{l=1}^{L_i} \text{Cat}(y_{il}|b_k) \quad (27.2)$

The “generative story” which this encodes is as follows: for document i , pick a topic q_i from π , call it k , and then for each word $l = 1: L_i$, pick a word from 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:

$$p(y_i, 1:R | q_i = k) = \prod_{r=1}^R \text{Cat}(y_{ir}|b_{kr}) \quad (27.3)$$

This is an unsupervised analog of naive Bayes classification.

We can also model count vectors. If the sum $L_i = v$ is known, we can use a multinomial:

$$p(n_i | L_i, q_i = k) = \text{Mu}(n_i | L_i, b_k) \quad (27.4)$$

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

$$p(n_i | q_i = k) = \prod_{v=1}^V \text{Poi}(n_{iv} | \lambda_{vk}) \quad (27.5)$$

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

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 $p(z_i) = N(\mu, \Sigma)$, where $z_i \in \mathbb{R}^K$, and a Gaussian likelihood of the form $p(y_i | z_i) = N(Wz_i, \sigma^2 I)$. 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

$$p(y_i|L_i|z_i) = \prod_{l=1}^L \text{Cat}(y_{il}|S(W_{zi})) \quad (27.6)$$

where $W \in RV \times K$ is a weight matrix and S is the softmax function. If we have a fixed number of categorical responses, we can use

$$p(y_i|R|z_i) = \prod_{r=1}^R \text{Cat}(y_{ir}|S(W_{rz_i})) \quad (27.7)$$

where $W_r \in RV \times K$ 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

$$p(n_i|L_i, z_i) = \text{Mu}(n_i|L_i, S(W_{zi})) \quad (27.8)$$

or a Poisson model

$$p(n_i|z_i) = \prod_{v=1}^V \text{Poi}(n_{iv}|\exp(w^T v, : z_i)) \quad (27.9)$$

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

$$p(y_i|L_i) = \prod_{l=1}^L p(y_{il}|z_i, W) N(z_i|\mu, \Sigma) dz_i \quad (27.10)$$

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 z_i and W . This can be regarded as a degenerate version of EM, that computes a point estimate of 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 data cases (Welling et al. 2008). A true EM algorithm would marginalize out the latent variables 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).

27.2.3 LDA and mPCA

In ePCA, the quantity W_{zi} 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 SK, the K-dimensional probability simplex. To avoid confusion with ePCA, we will denote such a latent vector by π_i . In this case, the natural prior for the latent variables is the Dirichlet, $\pi_i \sim \text{Dir}(\alpha)$. Typically we set $\alpha = \alpha_1 K$. If we set $\alpha_1 = 1$, we encourage π_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

$$p(n_i | L_i, \pi_i) = \text{Mu}(n_i | L_i, B\pi_i) \quad (27.11)$$

This model is called multinomial PCA or mPCA (Buntine 2002; Buntine and Jakulin 2004, 2006). See Figure 27.1(b). Since we are assuming $n_{iv} = k b v_k \pi_{iv}$, 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 $0 \leq b v, k \leq 1$ and $v b v, k = 1$. The corresponding marginal distribution has the form

$$p(n_i | L_i) = \text{Mu}(n_i | L_i, B\pi_i) \text{Dir}(\pi_i | \alpha) d\pi_i \quad (27.12)$$

Unfortunately, this integral cannot be computed analytically.

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

$$p(y_i, 1:L_i | \pi_i) = \prod_{l=1}^{L_i} \text{Cat}(y_{il} | B\pi_i) \quad (27.13)$$

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 π_{ik} are non-negative and sum to one. By contrast, in LSA, z_{ik} can be negative which makes interpretation 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 π_i for each document (similar to ePCA), rather than integrating it out. Thus in PLSI, there is no prior for π_i .

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

$$p(y_i, 1:R | \pi) = R \prod_{r=1}^R \text{Cat}(y_{ir} | B(r)\pi_i) \quad (27.14)$$

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 $z+ i$. This can be ensured by using a prior of the form

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

The likelihood is given by

$$p(n|z+ i) = \prod_{v=1}^V \text{Poi}(n_{iv} | b_T v z_{+i}) \quad (27.16)$$

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 $z+ i$.

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 $z+ i$ to be sparse, we can modify the prior to be a spike-and-Gamma type prior as follows:

$$p(z+ ik) = p_k I(z_{ik} = 0) + (1 - p_k) \text{Ga}(z_{ik} | \alpha_k, \beta_k) \quad (27.17)$$

where p_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.

已思考 44s

27.2 离散数据的分布式状态潜在变量模型

在本节中，我们总结了多种可行方法，用来构造如下形式的模型：**用于词袋 (bag of tokens) 的 $p(y_{i,1:L_i})$** 、**用于固定长度 token 向量的 $p(y_{1:R})$** 、以及**用于整数计数向量的 $p(n_i)$** 。

27.2.1 混合模型 (Mixture models) ——原文全译

最简单的方法是使用**有限混合模型**（第 11 章）。这会给每篇文档关联一个**离散潜在变量** $q_i \in \{1, \dots, K\}$ ，其中 K 是簇 (cluster) 的个数。我们使用离散先验 $q_i \sim \text{Cat}(\pi)$ 。对于**变长文档**，我们可以定义 $p(y_{il} | q_i = k) = b_{kv}$ ，其中 b_{kv} 是簇 k 生成词 v 的概率。 q_i 的取值被称为**主题 (topic)**，向量 b_k 是第 k 个主题的**词分布**。也就是说，似然具有如下形式

$$p(y_{i,1:L_i} | q_i = k) = \prod_{l=1}^{L_i} \text{Cat}(y_{il} | b_k) \quad (27.1)$$

对可见数据诱导出的分布为

$$p(y_{i,1:L_i}) = \sum_k \pi_k \prod_{l=1}^{L_i} \text{Cat}(y_{il} | b_k) \quad (27.2)$$

这段话编码的“**生成故事**”如下：对文档 i ，先从 π 选一个主题 q_i （记为 k ），然后对每个词位 $l = 1:L_i$ ，从 b_k 选一个词。我们会在本章后面讨论更复杂的生成模型。

如果我们拥有一组**固定的类别观测**，可以对每个输出变量使用不同的主题矩阵：

$$p(y_{i,1:R} | q_i = k) = \prod_{r=1}^R \text{Cat}(y_{ir} | b_k^{(r)}) \quad (27.3)$$

这是一种**无监督版本**的朴素贝叶斯分类。

我们也可以**建模计数向量**。如果和 $L_i = \sum_v n_{iv}$ 已知，就用**多项式分布**：

$$p(n_i | L_i, q_i = k) = \text{Mu}(n_i | L_i, b_k) \quad (27.4)$$

如果和未知，就用**条件在簇上的泊松密度**：

$$p(n_i | q_i = k) = \prod_{v=1}^V \text{Poi}(n_{iv} | \lambda_{vk}) \quad (27.5)$$

此时， $L_i | q_i = k \sim \text{Po}(\sum_v \lambda_{vk})$ 。

通俗解释

- 思想：**一篇文档只属于一个簇/主题；主题决定整篇里每个词出现的概率。

- (27.1)(27.2): 先抽主题 k , 再对每个词位独立地按 b_k 抽词。
 - (27.3): 当数据是**多个固定栏位** (如问卷多题) 时, 每个栏位各自有一个“该主题下的选项分布” $b_k^{(r)}$ 。
 - (27.4): 如果只看**词袋计数**且总词数 L_i 已知, 用**多项式**最自然。
 - (27.5): 若不知道总词数, 就假设每个词的计数是**泊松**; 总词数就变成若干泊松变量的和 (仍是泊松)。
 - **局限:** 每篇文档**只有一个主题** (硬分配), 表达力弱, 无法表示“体育+财经”的混合。
-

27.2.2 指数族 PCA (ePCA) ——原文全译

有限混合模型的表达能力很有限。更灵活的方法是使用**实值向量**潜在变量, 类似第 12 章的**因子分析 (FA)** 和**PCA**。在 PCA 中, 我们用高斯先验 $p(z_i) = N(\mu, \Sigma)$ ($z_i \in \mathbb{R}^K$) , 以及高斯似然 $p(y_i | z_i) = N(Wz_i, \sigma^2 I)$ 。该方法当然也能应用于离散或计数数据。实际上, 称为**潜在语义分析/索引 (LSA/LSI)**的方法 (Deerwester et al., 1990; Dumais & Landauer, 1997) , **正是**对“词×文档计数矩阵”做 PCA。

更适合**类别数据**的方法是使用**多项伯努利 (multinoulli)** /**多项式分布**。我们只需把**似然**改为

$$p(y_{i,1:L_i} | z_i) = \prod_{l=1}^{L_i} \text{Cat}(y_{il} | S(Wz_i)) \quad (27.6)$$

其中 $W \in \mathbb{R}^{V \times K}$ 是权重矩阵, S 是 softmax。若我们有固定数量的类别响应:

$$p(y_{1:R} | z_i) = \prod_{r=1}^R \text{Cat}(y_{ir} | S(W_r z_i)) \quad (27.7)$$

其中 $W_r \in \mathbb{R}^{V \times K}$ 是第 r 个响应变量的权重。该模型称为**分类 PCA (categorical PCA)** , 见图 27.1(a); 详见第 12.4 节。若是**计数**, 可用多项式

$$p(n_i | L_i, z_i) = \text{Mu}(n_i | L_i, S(Wz_i)) \quad (27.8)$$

或**泊松**

$$p(n_i | z_i) = \prod_{v=1}^V \text{Poi}(n_{iv} | \exp(w_{v,:}^\top z_i)) \quad (27.9)$$

以上模型都属于**指数族 PCA (ePCA)** (Collins et al., 2002; Mohamed et al., 2008) , 它是**无监督版**的 GLM。相应地, 可见变量上的边缘分布为

$$p(y_{i,1:L_i}) = \int_{\mathbb{R}^{L_i}} \left[\prod_{l=1}^{L_i} p(y_{il} | z_i, W) \right] N(z_i | \mu, \Sigma) dz_i \quad (27.10)$$

拟合很棘手，因为不共轭。 (Collins et al., 2002) 提出了交替坐标上升：在 z_i 与 W 之间交替估计。这可视为一种**退化的 EM**：在 E 步只对 z_i 做点估计。问题是这种退化方法**很容易过拟合**，因为潜在变量的数量与数据个案数同量级 (Welling et al., 2008)。真正的 EM 会把 z_i **积分掉**。对分类 PCA，可用**变分 EM**（见第 12.4 节）。更一般的模型可用 MCMC (Mohamed et al., 2008)。

通俗解释

- **改进点：**允许每篇文档有一个**连续向量** z_i （可以正负），经 W 映射并 softmax 得到“每个词的概率”。比“单一主题”更灵活。
 - **计数版：**总词数已知→多项式(27.8)；未知→泊松(27.9)（用 $\exp(\cdot)$ 确保强度为正）。
 - **技术难点：**后验没有解析形式（不共轭），所以要用变分或采样；用点估计的“准 EM”容易过拟合。
-

27.2.3 LDA 与 mPCA——原文全译

在 ePCA 里， Wz_i 表示指数族的**自然参数**。有时使用**对偶参数**更方便。对**多项式**而言，对偶参数是**概率向量**，而自然参数是**对数几率向量**。

如果想用对偶参数，就要**约束潜在向量**位于相应的参数空间。对**类别数据**，潜在向量应位于**概率单纯形** S_K 。为避免与 ePCA 混淆，我们把这类潜在向量记作 π_i 。此时，潜在变量的自然先验是 Dirichlet： $\pi_i \sim \text{Dir}(\alpha)$ 。通常设 $\alpha = \alpha \mathbf{1}_K$ 。当 $\alpha \ll 1$ 时，我们鼓励 π_i 稀疏（见图 2.14）。

当我们有**总和已知**的计数向量时，似然为

$$p(n_i | L_i, \pi_i) = \text{Mu}(n_i | L_i, B\pi_i) \quad (27.11)$$

该模型称为 **mPCA (multinomial PCA)** (Buntine, 2002; Buntine & Jakulin, 2004, 2006)，见图 27.1(b)。由于我们假设 $n_{iv} = \sum_k b_{vk}\pi_{ik}$ ，这可以被看作对计数矩阵的一种**矩阵分解**。注意我们用 $b_{v,k}$ 而非 $w_{v,k}$ 表参数，是因为我们施加了约束 $0 \leq b_{v,k} \leq 1$ 且 $\sum_v b_{v,k} = 1$ 。相应的边缘分布为

$$p(n_i | L_i) = \int \text{Mu}(n_i | L_i, B\pi_i) \text{Dir}(\pi_i | \alpha) d\pi_i \quad (27.12)$$

可惜此积分**无法解析求出**。

如果我们有**变长序列**（长度已知），可用

$$p(y_{i,1:L_i} \mid \pi_i) = \prod_{l=1}^{L_i} \text{Cat}(y_{il} \mid B\pi_i) \quad (27.13)$$

这称为 **LDA (Latent Dirichlet Allocation)** (Blei et al., 2003), 稍后会更详细介绍。LDA 可被看做 LSA 的**概率化扩展**, 其中 π_{ik} **非负且和为 1**; 相比之下, LSA 中 z_{ik} 可能为负, 解释性较差。

LDA 的前身 **PLSI** (Hofmann, 1999) 使用相同模型, 但对每个文档的 π_i 做**点估计** (类似 ePCA), 而不是积分掉。因此 PLSI **没有 π_i 的先验**。

我们可以修改 LDA, 以处理**固定数量**的不同类别响应:

$$p(y_{i,1:R} \mid \pi_i) = \prod_{r=1}^R \text{Cat}(y_{ir} \mid B^{(r)}\pi_i) \quad (27.14)$$

这被称为 **URP (user rating profile) 模型** (Marlin, 2003), 以及**simplex factor model** (Bhattacharya & Dunson, 2011)。

通俗解释

- **ePCA vs LDA/mPCA:**
 - ePCA: 潜在向量 z_i **不受非负与和为 1 的约束** (解释性较弱)。
 - LDA/mPCA: 潜在权重 π_i **非负且和为 1** (更像“主题占比”)。
- **mPCA (27.11)**: 总词数已知, 用 π_i 的凸组合 $B\pi_i$ 作为每个词的概率; $B = [b_{:,k}]$ 的每一列是一个主题的**词分布**。这等价于把计数矩阵做**非负/概率因子分解**。
- **LDA (27.13)**: 对每个词位都用同一个 π_i 混合出词分布 (也可扩展为词位级别的主题变量, 这是后文经典的 LDA 生成过程)。
- **PLSI**: 不放先验、对 π_i 做点估计, 容易过拟合、泛化差。
- $\alpha \ll 1$: 鼓励每篇文档只用**少数主题** (稀疏)。

小注: 原文在 (27.3)、(27.14) 用了 y_{il} 的下标, 按语境应为 y_{ir} (第 i 个样本的第 r 个响应)。

27.2.4 GaP 模型与非负矩阵分解 (NMF) ——原文全译

现在考虑**计数向量**, 并且我们**不约束**其总和被观测到。此时, 只需保证潜在变量**非负**, 用 z_i^+ 表示。这可由如下先验确保:

$$p(z_i^+) = \prod_{k=1}^K \text{Ga}(z_{ik}^+ \mid \alpha_k, \beta_k) \quad (27.15)$$

似然为

$$p(n_i \mid z_i^+) = \prod_{v=1}^V \text{Poi}(n_{iv} \mid b_{v,:}^\top z_i^+) \quad (27.16)$$

这称为 **GaP (Gamma–Poisson) 模型** (Canny, 2004)，见图 27.2(a)。

在 Buntine & Jakulin (2006) 中表明：当条件于固定 L_i 时，GaP 会化简为 mPCA。这源于这样一个事实：一组独立泊松随机变量在给定其和的条件下，会变成一个多项式分布（见如 Ross, 1989）。

如果在 GaP 中令 $\alpha_k = \beta_k = 0$ ，就得到称为 **NMF (非负矩阵分解)** 的方法 (Lee & Seung, 2001)，正如 Buntine & Jakulin (2006) 所示。NMF 不是一个概率生成模型，因为它没有给 z_i^+ 指定一个恰当的先验。此外，(Lee & Seung, 2001) 提出的算法是另一种退化 EM，因此会有过拟合问题。Buntine & Jakulin (2006) 给出了一些拟合 GaP 的过程，可克服这些问题。

为了鼓励 z_i^+ 稀疏，我们可以把先验改成***“尖峰+Gamma (spike-and-Gamma) ***”：

$$p(z_{ik}^+) = \rho_k \mathbf{1}(z_{ik}^+ = 0) + (1 - \rho_k) \text{Ga}(z_{ik}^+ \mid \alpha_k, \beta_k) \quad (27.17)$$

其中 ρ_k 是在 0 处的“尖峰”概率。这称为**条件 Gamma–Poisson 模型** (Buntine & Jakulin, 2006)。对这种先验，修改吉布斯采样很简单，不过我们这里不展开。

通俗解释

- **GaP:**
 - 潜在因子 z_i^+ 非负，每个词的期望计数由“主题强度”线性组合给出，再进入泊松分布。
 - 若把总词数 L_i 固定，泊松→多项式，于是就退化为 mPCA。
- **NMF:**
 - 令 $\alpha_k = \beta_k = 0$ 相当于**不放先验**，得到经典的 NMF (只做非负因子分解)。
 - 算法本质像**退化 EM**，易过拟合、缺概率解释。
- **稀疏先验**：用“在 0 处有质量”的先验 (spike) + Gamma，使很多 z_{ik}^+ 变 0，得到**更可解释的少数激活主题**。

小结要点 (记忆版)

- **三类观测形式**：
 - $p(y_{i,1:L_i})$: 词序/词袋； $p(y_{1:R})$: 固定多栏位类别； $p(n_i)$: 计数向量。

- **混合模型**: 每文档单一主题 q_i ; 已知总词数→多项式; 未知→泊松。简单但**表达力弱**。
- ePCA: 连续潜在向量 z_i (无非负/归一约束), 用 softmax + (多项式/泊松) 建模类别/计数; **不共轭**→需变分/MCMC; 点估计式“准 EM”**易过拟合**。
- mPCA/LDA (**对偶参数**): 潜在权重 π_i 位于**单纯形**, Dirichlet 先验;
 - mPCA: $p(n_i | L_i, \pi_i) = \text{Mu}(n_i | L_i, B\pi_i)$, 等价**概率矩阵分解**;
 - LDA: $p(y | \pi_i) = \prod \text{Cat}(y_{il} | B\pi_i)$, LSA 的**概率扩展**;
 - PLSI: 对 π_i 做点估计、**无先验** (易过拟合)。
- GaP/NMF: 不固定总和时用 GaP (Gamma 先验 + 泊松似然); 条件于总和→回到 mPCA; $\alpha = \beta = 0$ 即 NMF (非概率、易过拟合)。
- **稀疏性**: $\alpha \ll 1$ (Dirichlet) 或 spike-and-Gamma (GaP) 都能促使**少量主题激活**, 更易解释。

细微排印提示: 原文中 (27.3)、(27.14) 的 y_{il} 按语境应为 y_{ir} ; “ $n_{iv} = \sum_k b_{vk}\pi_{iv}$ ”更合理的理解是**期望**由该线性组合给出, 且下标应为 π_{ik} 。

这一节主要在讲什么? 我不太懂

已思考 11s

没关系, 这一节信息量很大。我用“鸟瞰+对比”的方式给你捋清楚它在讲啥。

一句话版

27.2 节是在给离散数据 (文本、问卷、计数等) 做“潜在因子建模”的方法地图。

它把常见思路分成四类: **混合模型**、ePCA、LDA/mPCA、GaP/NMF, 各自适配不同的数据形式与约束 (是否需要多主题混合、是否已知总词数、参数是否非负/归一等)。

四类方法直观图 (比喻版)

把一篇文档想成“**一盘菜**”, 词就是“**食材**”。

1. 混合模型 (Mixture) :

这盘菜只属于**一种菜系** (一个主题)。主题决定整盘菜里**所有食材的比例**。

优点: 简单; 缺点: **一盘菜只能一种菜系**, 表达弱。

2. ePCA (指数族 PCA) :

给每盘菜一个**连续向量 z**, 通过一层变换 (比如 softmax) 得到“**食材概率**”。

优点：很灵活；缺点：**推断难（不共轭）**，常要变分/MCMC；用“点估计”容易过拟合。

3. LDA / mPCA (用对偶参数：概率向量 π) :

直接让“菜系配比” π **非负且总和为1**，有 Dirichlet 先验；每盘菜可由**多种菜系混合**。

- mPCA：已知总食材数（总词数 L ）→多项式建模；像**概率版的矩阵分解**。
- LDA：对序列/词袋逐词抽样， π 是该盘菜的**多菜系配比**。

优点：**可解释、稳健**；PLSI 是它的“无先验点估计版”，更易过拟合。

4. GaP / NMF (非负强度 + 泊松) :

不限定总食材数。让“主题强度” z^+ **非负**，每种食材的期望=强度的非负组合，再用**泊松**产生成数。

- 条件在总数固定 \Rightarrow 退化回 mPCA。
- 把先验“关掉” \Rightarrow NMF（不是严格的概率模型，易过拟合）。
- 想更稀疏 \Rightarrow 用“0 点尖峰 + Gamma”先验。

放到一张对照表

方法	潜在变量	数据假设	多主题混合	训练与注意点	适用场景
混合模型	q (离散：1个主题/文档)	类别/计数；总数已知→多项式；未知→泊松	否 (每文档1主题)	简单；表达力弱	明显分簇、粗粒度分组
ePCA	$z \in \mathbb{R}^K$ (可正负)	类别/计数 (用 softmax、多项式/泊松)	是 (隐式)	不共轭，需变分/MCMC；点估计易过拟合	需要连续嵌入、灵活拟合
mPCA	$\pi \in \Delta^K$ (非负、和为1)	计数且总数已知 (多项式)	是	积分不可解析；用近似推断	词袋计数、想要“主题占比”解释
LDA	$\pi \in \Delta^K$	词序/词袋；逐词类别抽样	是	经典近似推断 (变分/采样)	文本主题模型

方法	潜在变量	数据假设	多主题混合	训练与注意点	适用场景
GaP	$z^+ \geq 0$	计数、总数不限 (泊松)	是	条件在总数 → mPCA; 可用稀疏先验	点击/计数流量、事件强度
NMF	$z^+ \geq 0$	矩阵因子分解视角	是	非概率; 易过拟合	快速分解、可解释分量

关键概念用大白话

- **自然参数 vs. 对偶参数:**

自然参数（如“对数几率”）适合用在 ePCA；对偶参数是**概率本身**（各主题占比），适合 LDA/mPCA。

- **单纯形 Δ^K :** 所有分量非负且**和为 1**，就是“占比向量”的家。

- **是否已知总数 L :**

- 已知 \Rightarrow 多项式 (mPCA) 。
- 未知 \Rightarrow 泊松 (GaP) 。

- **过拟合的来源:**

给每个样本都单独一个“自由”的潜在向量并用**点估计**（如 PLSI、退化 EM）时，很容易把噪音当成结构。放先验、做积分（边缘化）/近似推断更稳。

实操选型小抄

- **只是做聚类**，每篇文档归到一个主题：用**混合模型**。
- **想要“多主题配比”、可解释的主题占比**：优先 LDA（序列/词袋）或 mPCA（已知总数的计数）。
- **计数总量浮动、像流量/事件计数**：用 GaP；要快但不严格概率化可尝试 NMF。
- **要更灵活的连续嵌入**、配合 GLM 思想：用 ePCA，但注意推断复杂度和过拟合。

要点小结

- 这一节就是：**给离散数据建潜在因子模型的四条主线**（混合、ePCA、LDA/mPCA、GaP/NMF），分别对应**不同的数据假设**（是否已知总数）、**不同的潜在变量约束**（是否非负/归一）、以及**不同的训练难度**（是否共轭、是否需要近似推断）。

- **混合模型简单但只能“一文档一主题”。**
- ePCA灵活但推断难，点估计易过拟合。
- LDA/mPCA用“概率向量”当潜在变量，**解释性强**（像主题占比），是实践常用派。
- GaP适合总数不固定的计数数据；NMF是它的非概率极限版；想要稀疏更可解释可用“尖峰+Gamma”先验。

如果你想，我可以用一个小的玩具数据，画出这几种方法学到的“主题/成分”长什么样，帮助直观对比。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

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, $q_i \in \{1, \dots, K\}$, drawn from a global distribution π . In LDA, every word is assigned to its own topic, $q_{il} \in \{1, \dots, K\}$, drawn from a document-specific distribution π_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:

$$\pi_i | \alpha \sim \text{Dir}(\alpha_1 K) \quad (27.18)$$

$$q_{il} | \pi_i \sim \text{Cat}(\pi_i) \quad (27.19)$$

$$b_k | \gamma \sim \text{Dir}(\gamma_1 V) \quad (27.20)$$

$$y_{il} | q_{il} = k, B \sim \text{Cat}(b_k) \quad (27.21)$$

This is illustrated in Figure 27.2(b). We can marginalize out the q_i variables, thereby creating a direct arc from π_i to y_{il} , with the following CPD:

$$p(y_{il} = v | \pi_i) = k p(y_{il} = v | q_{il} = k) p(q_{il} = k) = k \pi_i b_k v \quad (27.22)$$

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 b_k defines a distribution over V words; each k is known as a topic. Each document vector π_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 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 important since in natural language, words can often have multiple meanings, a phenomenon 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 $p(q_{il} = k | \pi_i, \theta)$, 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 x_i , we can infer the topic distribution π_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 π_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 it is crucial that π_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 interpretation 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 p is defined as

$$\text{perplexity}(p, q) = 2H(p, q) \quad (27.23)$$

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

$$H(p, q) = \lim_{N \rightarrow \infty} -\frac{1}{N} \sum_{y=1}^N \log p(y|q) \quad (27.24)$$

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 $y^* \in \mathcal{Y}$. (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

$$p_{\text{emp}}(y^*) = \delta_{y^*} \quad (27.25)$$

In this case, the cross-entropy becomes
 $H(p_{emp}, q) = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^{L_i} \log q(y_{ij} | \mathbf{y}_{1:i-1})$ (27.26)
 and the perplexity becomes
 $\text{perplexity}(p_{emp}, q) = \exp(-H(p_{emp}, q)) = \exp\left(-\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^{L_i} \log q(y_{ij} | \mathbf{y}_{1:i-1})\right)$ (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

$$H = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^{L_i} \log q(y_{ij})$$
 (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

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

Intuitively, perplexity measures 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 $p(y_{ij} | \mathbf{y}_{1:i-1}) = 1/K$. Then the perplexity is $(1/K)^N = K$. 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 B (e.g., the posterior mean estimate) and approximately integrating out 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 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.

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:

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

$$p(\pi_i | \cdot) = \text{Dir}(\{\alpha_k + 1 | (z_{il} = k)\}) \quad (27.31)$$

$$p(b_k | \cdot) = \text{Dir}(\{\gamma_v + 1 | (x_{il} = v, z_{il} = k)\}) \quad (27.32)$$

However, one can get better performance by analytically integrating out the π_i 's and the 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 $c_{ivk} = \sum_l l=1 \mathbb{I}(q_{il} = k, y_{il} = v)$ 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

$$p(q | \alpha) = \prod_i \prod_k \text{Cat}(q_{il} | \pi_i) \text{Dir}(\pi_i | \alpha K) d\pi_i \quad (27.33)$$

$$= \Gamma(K\alpha) \prod_i \Gamma(\alpha) \prod_k \Gamma(N_i) \prod_{l=1}^{N_i} \Gamma(c_{ik} + \alpha) \prod_{l=1}^{N_i} \Gamma(L_i + K\alpha) \quad (27.34)$$

By similar reasoning, one can show

$$p(y | q, \gamma) = \prod_i \prod_k \text{Cat}(y_{il} | b_{k,xil}) \prod_k \text{Dir}(b_k | \gamma V) db_k \quad (27.35)$$

$$= \Gamma(V\beta) \prod_i \Gamma(\beta) \prod_k \Gamma(K) \prod_{l=1}^{N_i} \Gamma(c_{vk} + \beta) \prod_{l=1}^{N_i} \Gamma(c_k + V\beta) \quad (27.36)$$

From the above equations, and using the fact that $\Gamma(x+1)/\Gamma(x) = x$, we can derive the full conditional for $p(q_{il} | q_{-i,l}, \gamma)$.

Define c_{-ivk} to be the same as c_{ivk} except it is computed by summing over all locations in document i except for q_{il} . Also, let $y_{il} = v$. Then

$$p(q_{il} | q_{-i,l}, y, \alpha, \gamma) \propto c_{-ivk} + \gamma c_{-ik} + V y_{il} c_{-ik} + \alpha L_i + K\alpha \quad (27.37)$$

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, $q_{il} \in \{1, \dots, K\}$. 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 $p(\text{money}|k=1) = p(\text{loan}|k=1) = p(\text{bank}|k=1) = 1/3$ 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 π_i and q_{il} 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

$$q(\pi_i, q_{il}) = \text{Dir}(\pi_i | \tilde{\pi}_i) \mid \text{Cat}(q_{il} | \tilde{q}_{il}) \quad (27.38)$$

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:

$$\tilde{q}_{il} \propto b_{il,k} \exp(E[\log \pi_{ik}]) \quad (27.39)$$

where

$$E[\log \pi_{ik}] = \psi_k(\tilde{\pi}_i) = \Psi(\tilde{\pi}_{ik}) - \Psi(k \tilde{\pi}_{ik}) \quad (27.40)$$

where Ψ is the digamma function. The update for $q(\pi_i)$ is obtained by adding up the expected counts:

$$\tilde{\pi}_{ik} = \alpha_k + \tilde{q}_{il} \quad (27.41)$$

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

$$\hat{b}_{vk} \propto \gamma v + N \sum_{i=1}^N \sum_{l=1}^{L_i} q_{il} k_l (y_{il} = v) \quad (27.42)$$

27.3.6.2 Count version

Note that the E step takes $O(NL^2V)$ 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 $p(\pi_i, c_i | n_i, L_i)$, where we write c_i as shorthand for $c_{i1} \dots c_{iL_i}$. We will again use a fully factorized (mean field) approximation of the form

$$q(\pi_i, c_i) = \text{Dir}(\pi_i | \tilde{\pi}_i) \times \text{Mu}(c_i | n_i, \tilde{c}_i) \quad (27.43)$$

The new E step becomes

$$\tilde{\pi}_{ik} = \alpha_k + v n_i \tilde{c}_{ivk} \quad (27.44)$$

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

The new M step becomes

$$\hat{b}_{vk} \propto \gamma v + i n_i \tilde{c}_{ivk} \quad (27.46)$$

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 $\gamma = 1$, VB will encourage 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

$$q(\pi_i, c_i, B) = \text{Dir}(\pi_i | \tilde{\pi}_i) \times \text{Mu}(c_i | n_i, \tilde{c}_i) \times \text{Dir}(B | \tilde{B}) \quad (27.47)$$

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

$$\tilde{c}_{ivk} \propto \exp(E[\log b_{vk}] + E[\log \tilde{\pi}_{ik}]) \quad (27.48)$$

Also, the M step becomes

$$\tilde{b}_{vk} = \gamma v + i \tilde{c}_{ivk} \quad (27.49)$$

No normalization is required, since we are just updating the pseudocounts.

27.3.7 Fitting using online variational inference

In the batch 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 B treating the expected sufficient statistics from the single data case as if the whole data set had those statistics. Finally, we make a partial update for the variational parameters for B , putting weight pt on the new estimate and weight $1 - pt$ on the old estimate. The step size pt 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 $\log p(D|K)$.
- Use non-parametric Bayesian methods (Teh et al. 2006).

已思考 17s

下面把整段内容逐句完整翻译，并在每一小节后给出通俗、详细的解释（尽量少用高深数学），最后再总结要点。原文里个别地方存在排版/符号不一致（如 x_{il} vs. y_{il} 、 z_{il} vs. q_{il} ），我会在解释里标注出来，意思以上下文为准。

27.3 潜在狄利克雷分配 (LDA)

在本节中，我们将详细解释潜在狄利克雷分配 (latent Dirichlet allocation, LDA; Blei et al., 2003) 模型。

27.3.1 基础

在多项伯努利混合模型 (mixture of multinoullis) 中，每篇文档被分配给一个单一主题， $q_i \in \{1, \dots, K\}$ ，它是从一个全局分布 π 中抽样得到的。在 LDA 中，每个词都会被分配给它自己的主题， $q_{il} \in \{1, \dots, K\}$ ，它是从文档特定的分布 π_i 中抽样得到的。由于一篇

文档属于的是“**对主题的一个分布**”，而不是单一主题，这个模型被称为**混合成员模型**（admixture / mixed membership model；Erosheva et al., 2004）。该模型在文本分析之外还有许多应用，例如遗传学（Pritchard et al., 2000）、健康科学（Erosheva et al., 2007）、社交网络分析（Airoldi et al., 2008）等。

给参数加上共轭先验后，完整模型如下：

$$\pi_i \mid \alpha \sim \text{Dir}(\alpha \mathbf{1}_K) \quad (27.18)$$

$$q_{il} \mid \pi_i \sim \text{Cat}(\pi_i) \quad (27.19)$$

$$b_k \mid \gamma \sim \text{Dir}(\gamma \mathbf{1}_V) \quad (27.20)$$

$$y_{il} \mid q_{il} = k, B \sim \text{Cat}(b_k) \quad (27.21)$$

如图 27.2(b) 所示。我们可以把 q_i 变量边缘化掉，于是会从 π_i 到 y_{il} 直接产生一条边，其条件概率分布（CPD）如下：

$$p(y_{il} = v \mid \pi_i) = \sum_k p(y_{il} = v \mid q_{il} = k) p(q_{il} = k) = \sum_k \pi_{ik} b_{kv} \quad (27.22)$$

正如我们在引言中提到的，这与（Buntine 2002）提出的**多项式 PCA (mPCA) 模型非常相似**，而 mPCA 又与分类 PCA（categorical PCA）、GaP、NMF 等密切相关。

LDA 有一个有趣的**几何解释**。每个向量 b_k 定义了在 V 个词上的一个分布；每个 k 被称为一个**主题**。每个文档向量 π_i 定义了在 K 个主题上的一个分布。因此我们把每个文档建模为“**按主题混合而成**”。等价地，我们也可以把 LDA 视为一种**降维**（通常 $K < V$ ）：我们把位于 V 维单纯形（一个归一化的文档计数向量 x_i ）上的点投影到 K 维单纯形上。图 27.3 做了示意：当 $V = 3$ 个词、 $K = 2$ 个主题时，观测到的文档（位于 3 维单纯形）被近似为位于由两个主题向量张成的 2 维单纯形上；每个主题向量也位于 3 维单纯形中。

把**单纯形**作为潜在空间而非欧氏空间的一个优势是：**单纯形能处理歧义**。这很重要，因为自然语言中词往往**一词多义**（polysemy）。例如“play”既可以是动词（如“to play ball”“to play the coronet”），也可以是名词（如“Shakespeare's play”）。在 LDA 中，我们可以有多个主题都能生成词“play”（见图 27.4），从而反映出这种歧义。

给定文档 i 的第 l 个词，我们可以计算 $p(q_{il} = k \mid y_i, \theta)$ ，从而推断它**最可能的主题**。把单个词孤立地看，可能很难分辨含义，但通过查看文档中的**其他词**就能消除歧义。具体地，给定 x_i （文档的词统计），我们可以推断该文档的主题分布 π_i ；它对判定 q_{il} 起到**先验作用**。图 27.5 展示了 TASA 语料库中的三个文档²。在第一篇里，有各种与音乐有关的词，这暗示 π_i 的质量会主要放在“音乐主题”（编号 77）上；这又会使“play”的**音乐意义**最可能（图中用上标标注）。第二篇把“play”解释为**戏剧**意义，第三篇解释为**体育**意义。注

意：将 π_i 设为**潜在变量**是至关重要的，这样信息可以在各个 q_{il} 之间流动，从而让**局部消歧**能利用整篇文档的全部词。

通俗解释

- 与“每篇只选一个主题”的混合模型不同，LDA 允许**同一篇文档里的不同词来自不同主题**；文档用 π_i 表示**主题配比**。
 - π_i 和每个主题的词分布 b_k 都放了**Dirichlet 先验**，方便推断（共轭）。
 - “单纯形”就是“各分量非负、和为 1”的空间，很适合表示“占比”。这让**多义词**能被不同主题解释。
-

27.3.2 主题的无监督发现

LDA 的主要目的之一，是在一个大型文档集合（语料库）中**发现主题**（例子见图 27.12）。不幸的是，由于模型是**不可辨识的**（unidentifiable），对主题的解释可能会很困难（Chang et al., 2009）。一种方法叫**有标签的 LDA (labeled LDA)**（Ramage et al., 2009），它利用文档上的**标签 (tags)** 来确保可辨识性。具体地，它**强制主题与标签一一对应**，然后为每个标签学习一个词的分布。这可让结果更容易解释。

通俗解释

- **不可辨识**：不同的参数设置可能产生几乎一样的观测分布，导致主题“转个身就变样”。给定**外部标签**能把主题“钉住”，更稳定。
-

27.3.3 把 LDA 当语言模型做定量评估

为了**定量评估** LDA，我们可以把它当作一个**语言模型**（对词序列的概率分布）。当然，它并不是一个很好的语言模型，因为它**忽略词序**、只看单词（unigram），但把它和其他基于 unigram 的模型（如多项伯努利混合、pLSI）比较仍然有意义。这类简单语言模型在**信息检索**中有时很有用。评估语言模型的**标准方法**是使用**困惑度 (perplexity)**，我们在下文定义。

27.3.3.1 困惑度

给定随机过程 p 与语言模型 q ，它们的**困惑度**定义为

$$\text{perplexity}(p, q) = 2^{H(p, q)} \quad (27.23)$$

其中 $H(p, q)$ 是两个随机过程的**交叉熵**，定义为

$$H(p, q) = \lim_{N \rightarrow \infty} -\frac{1}{N} \sum_{y_{1:N}} p(y_{1:N}) \log q(y_{1:N}) \quad (27.24)$$

当 $q = p$ 时，交叉熵（因此困惑度）最小；这时模型的预测与“真实分布”一样好。

我们可以用一条**很长的测试序列**（由多篇文档和多句构成，包含句末标记），记作 $y_{1:N}^*$ ，来近似这个随机过程（只要过程是**平稳且遍历的**，序列越长近似越准；Cover & Thomas, 2006）。定义经验分布（过程的近似）为

$$p_{\text{emp}}(y_{1:N}) = \delta_{y_{1:N}^*}(y_{1:N}) \quad (27.25)$$

在这种情况下，交叉熵变为

$$H(p_{\text{emp}}, q) = -\frac{1}{N} \log q(y_{1:N}^*) \quad (27.26)$$

困惑度变为

$$\text{perplexity}(p_{\text{emp}}, q) = 2^{H(p_{\text{emp}}, q)} = q(y_{1:N}^*)^{-1/N} = \prod_{i=1}^N \frac{1}{q(y_i^* | y_{1:i-1}^*)} \quad (27.27)$$

可见这就是**逆预测概率的几何平均数**，这是困惑度的常见定义（Jurafsky & Martin, 2008, p96）。

对于**unigram 模型**，交叉熵项为

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

其中 N 是文档数， L_i 是第 i 篇文档的词数。因此模型 q 的困惑度为

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

直觉上，困惑度衡量的是模型预测分布的**加权平均分支因子**。假设模型预测每个符号等可能，即 $p(y_i | y_{1:i-1}) = 1/K$ 。则困惑度为 $((1/K)^N)^{-1/N} = K$ 。如果有些符号更常见，而模型也准确反映了这一点，它的困惑度就会**低于** K 。当然 $H(p, p) = H(p) \leq H(p, q)$ ，所以困惑度**不可能低于**底层随机过程的熵。

27.3.3.2 LDA 的困惑度

关键量是 $p(v)$ ，即模型对可能词的**预测分布**（隐式地以训练集为条件）。对 LDA，可通过代入 B （例如后验均值估计），并用**均值场**对 q 做近似积分来得到（更准确的预测似然近似见 Wallach et al., 2009）。

在图 27.6 中，我们把 LDA 与其他若干简单 unigram 模型比较：分别是**多项伯努利的 MAP、多项伯努利混合的 MAP、以及 pLSI**。（进行 MAP 时， B 上使用的 Dirichlet 先验

与 LDA 相同。) 指标是等式 (27.29) 的困惑度, 数据是 TREC AP 语料的一个子集: 包含 16,333 篇新闻电文、23,075 个唯一词。可见 LDA 的表现显著优于这些方法。

通俗解释

- 困惑度越低越好; 它相当于“平均每步需要面对的有效分支数”。能把常见词预测得更高、更准的模型, 困惑度就更低。
 - 在同等先验/设定下, LDA 往往比“每文档单主题”或 pLSI 这类简单模型更会“猜词”。
-

27.3.4 用(折叠) Gibbs 采样拟合

对 LDA 推导 Gibbs 采样算法是直接的。完整条件分布如下:

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

$$p(\pi_i \mid \cdot) = \text{Dir}\left(\{\alpha_k + \sum_l \mathbf{I}(z_{il} = k)\}\right) \quad (27.31)$$

$$p(b_k \mid \cdot) = \text{Dir}\left(\{\gamma_v + \sum_{i,l} \mathbf{I}(x_{il} = v, z_{il} = k)\}\right) \quad (27.32)$$

不过, 通过把 π_i 和 b_k (它们都服从 Dirichlet) **解析地积分掉**, 只对离散的 q_{il} 采样, 能得到更好的表现。这一方法最早由 Griffiths & Steyvers (2004) 提出, 是**折叠 Gibbs 采样** (collapsed Gibbs sampling) 的一个例子。图 27.7(b) 显示此时所有 q_{il} 变量是**完全相关的**。不过我们仍可逐个进行采样, 如下所述。

先引入一些记号。令

$$c_{ivk} = \sum_{l=1}^{L_i} \mathbf{I}(q_{il} = k, y_{il} = v)$$

表示词 v 在文档 i 中被分配到主题 k 的次数。令

$$c_{ik} = \sum_v c_{ivk}$$

表示文档 i 中任何词被分配到主题 k 的次数。令

$$c_{vk} = \sum_i c_{ivk}$$

表示词 v 在所有文档中被分到主题 k 的次数。令

$$n_{iv} = \sum_k c_{ivk}$$

为词 v 在文档 i 中出现的次数；这是观测到的。令

$$c_k = \sum_v c_{vk}$$

为被分配到主题 k 的词的总数。最后，令

$$L_i = \sum_k c_{ik}$$

为文档 i 的总词数；这也是观测到的。

现在可以推导边缘先验。由等式 5.24 可得

$$p(q \mid \alpha) = \prod_i \int [\prod_{l=1}^{L_i} \text{Cat}(q_{il} \mid \pi_i)] \text{Dir}(\pi_i \mid \alpha \mathbf{1}_K) d\pi_i \quad (27.33)$$

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

用同样的思路可得

$$p(y \mid q, \gamma) = \prod_k \int [\prod_{i,l: q_{il}=k} \text{Cat}(y_{il} \mid b_k)] \text{Dir}(b_k \mid \gamma \mathbf{1}_V) db_k \quad (27.35)$$

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

由以上等式并利用 $\Gamma(x+1)/\Gamma(x) = x$ ，可推出条件分布 $p(q_{il} \mid q_{-i,l})$ 。定义 c_{ivk}^- 与 c_{ivk} 相同，但不包含当前位置 (i, l) 的计数；并设 $y_{il} = v$ 。则

$$p(q_{i,l} = k \mid q_{-i,l}, y, \alpha, \gamma) \propto \frac{c_{v,k}^- + \gamma}{c_k^- + V\gamma} \cdot \frac{c_{i,k}^- + \alpha}{L_i + K\alpha} \quad (27.37)$$

可见，某个词在某个文档中被分配给某主题，既取决于该主题生成这个词的频繁程度（第一项），也取决于该主题在该文档中被使用的频繁程度（第二项）。

给定式 (27.37)，折叠 Gibbs 的实现如下：先随机给每个词赋一个主题 $q_{il} \in \{1, \dots, K\}$ 。然后对语料中的每个词，按当前主题把相关计数减去，根据 (27.37) 抽取新主题，再把计数加回去；重复。由于计数矩阵非常稀疏，这个算法可以做得很高效。

说明：上面 (27.30)–(27.32) 的原文把 y_{il} 写成了 x_{il} ，把 q_{il} 写成了 z_{il} ，是记号不统一；此处已按上下文解释。

通俗解释

- 折叠 Gibbs 的“妙处”：把连续的 π_i, b_k 积分掉，只采样“每个词属于哪个主题”，更快更稳。
 - 更新规则很好记：(词在该主题里常见) \times (该文档常用该主题) \rightarrow 概率就高。
-

27.3.5 示例

图 27.8 用一个拥有**两个主题、五个词**的小例子演示了这一过程。左图展示了从 LDA 采样出的 16 篇文档，其中

$$p(\text{money} \mid k = 1) = p(\text{loan} \mid k = 1) = p(\text{bank} \mid k = 1) = 1/3,$$

$$p(\text{river} \mid k = 2) = p(\text{stream} \mid k = 2) = p(\text{bank} \mid k = 2) = 1/3.$$

例如，第一篇文档包含了 4 次 “bank”（在 “bank” 列第 1 行的四个点表示），以及若干金融术语。右图显示了 Gibbs 采样 64 次迭代后的状态。多数情况下，“正确”的主题已经被分配给了每个 token。比如在文档 1 中，词 “bank” 被正确分到**金融主题**，这是由 “money”“loan” 的出现所暗示的。参数的**后验均值估计**为

$$\hat{p}(\text{money} \mid k = 1) = 0.32, \hat{p}(\text{loan} \mid k = 1) = 0.29, \hat{p}(\text{bank} \mid k = 1) = 0.39,$$

$$\hat{p}(\text{river} \mid k = 2) = 0.25, \hat{p}(\text{stream} \mid k = 2) = 0.40, \hat{p}(\text{bank} \mid k = 2) = 0.35,$$

考虑到只有 16 个训练样本，这个精度已经很可观。

通俗解释

- “bank” 一词一词多义（银行/河岸）。在带有“money/loan”的上下文里，它就被判为“银行”意义；在带有“river/stream”的上下文里，则判为“河岸”。
-

27.3.6 用批量变分推断拟合

比 MCMC 更快的一个替代方法是使用**变分 EM**。（无法用**精确 EM**，因为对 π_i 与 q_i 的精确推断是**不可 tractable** 的。）细节如下。

27.3.6.1 序列版本

按照 Blei et al. (2003)，我们使用**完全因子化（均值场）**的近似：

$$q(\pi_i, q_i) = \text{Dir}(\pi_i \mid \tilde{\pi}_i) \prod_l \text{Cat}(q_{il} \mid \tilde{q}_{il}) \quad (27.38)$$

沿用均值场套路。对 $q(q_{il})$ ，用贝叶斯法则，但需要对先验取期望：

$$\tilde{q}_{ilk} \propto \tilde{b}_{y_i, l, k} \exp(\text{E}[\log \pi_{ik}]) \quad (27.39)$$

其中

$$\text{E}[\log \pi_{ik}] = \psi_k(\tilde{\pi}_{i \cdot}) = \Psi(\tilde{\pi}_{ik}) - \Psi(\sum_k \tilde{\pi}_{ik}) \quad (27.40)$$

Ψ 为 digamma 函数。对 $q(\pi_i)$ 的更新通过累加期望计数得到：

$$\tilde{\pi}_{ik} = \alpha_k + \sum_l \tilde{q}_{ilk} \quad (27.41)$$

M 步通过累加期望计数再归一化得到：

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

27.3.6.2 计数版本

注意：E 步需要 $O((\sum_i L_i) V K)$ 的空间去存 \tilde{q}_{ilk} 。在 mPCA (计数) 版本里做推断要省空间得多，只需 $O(N V K)$ (当文档很长时差别巨大)。相比之下，折叠 Gibbs 必须显式操作 q_{il} 。

我们聚焦于近似 $p(\pi_i, c_i \mid n_i, L_i)$ (其中 c_i 代表所有 c_{ivk} 的集合)。仍用完全因子化的近似：

$$q(\pi_i, c_i) = \text{Dir}(\pi_i \mid \tilde{\pi}_i) \prod_v \text{Mu}(c_{iv \cdot} \mid n_{iv}, \tilde{c}_{iv \cdot}) \quad (27.43)$$

新的 E 步：

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

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

新的 M 步：

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

27.3.6.3 VB 版本

我们现在把算法从 EM 改为 VB (变分贝叶斯)，从而同时推断参数与潜在变量。好处有二：其一，通过设置 $\gamma \ll 1$ ，VB 会鼓励 B 稀疏 (参见 21.6.1.6 节)；其二，便于推广到在线学习 (下节)。

新的后验近似为

$$q(\pi_i, c_i, B) = \prod_v \text{Dir}(\pi_i | \tilde{\pi}_i) \prod_v \text{Mu}(c_{iv} | n_{iv}, \tilde{c}_{iv}) \prod_k \text{Dir}(b_{\cdot k} | \tilde{b}_{\cdot k}) \quad (27.47)$$

\tilde{c}_{ivk} 的更新变为

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

同时，M 步为

$$\tilde{b}_{vk} = \gamma_v + \sum_i \tilde{c}_{ivk} \quad (27.49)$$

这里无需归一化，因为我们只是更新伪计数。

通俗解释

- **序列版**: 对“每个词的主题”存一个分布 \tilde{q}_{il} ，空间大。
 - **计数版**: 把“同词的多个 token”合在一起，用 n_{iv} 次“把蛋糕分给各主题”的方式，空间省很多。
 - **VB**: 连参数 B 也用后验 (Dirichlet) 来估计；调 γ 可让主题更稀疏、可解释。
-

27.3.7 用在线变分推断拟合

在批量版本中，E 步显然需要 $O(NKVT)$ 的时间 (T 是均值场迭代次数，通常 $T \approx 5$)。当文档很多时会很慢。可以用随机梯度 (§8.5.2) 来做在线变分推断，如下。

按 (Hoffman et al., 2010) 可推导在线版本：先按常规做一个 E 步；然后把单个数据产生的期望充分统计当作“整个数据集都长这样”，来计算 B 的变分参数；最后对 B 的变分参数做一次部分更新：用权重 ρ_t 放在新的估计、 $1 - \rho_t$ 放在旧的估计上； ρ_t 随时间按式 (8.83) 递减。整体算法在算法 3 中总结。实践中应使用小批量 (mini-batch)，详见 §8.5.2.3。 (Hoffman et al., 2010) 使用了 256–4096 的 batch 大小。

图 27.9 画出了测试集大小为 1000 的 perplexity 与“已处理文档数 (E 步次数)”的关系 (数据来自英文维基)。图示：在线变分比离线变分快得多，而结果相近。

通俗解释

- 在线 VI 就是“边看边学”：每次只看一小批文档，用保守的学习率缓慢更新主题；速度快、可扩展。
-

27.3.8 如何确定主题数 K

选择主题数 K 是一个标准的**模型选择**问题。常见做法：

- **用退火重要性采样** (§24.6.2) 近似证据 (Wallach et al., 2009)。
- **交叉验证**：用测试集 log-likelihood。
- **用变分下界**作为 $\log p(D | K)$ 的代理。
- **用非参数贝叶斯方法** (Teh et al., 2006)。

通俗解释

- 没有“万能的 K ”。可以靠验证集、近似证据，或直接上**可变维**的非参模型（如 HDP-LDA）来让数据自己“决定”主题数。
-

要点小结 (便于回忆)

- **LDA 的核心**：
文档有一个**主题配比** $\pi_i \sim \text{Dir}$, 每个词有一个**主题指派** $q_{il} \sim \text{Cat}(\pi_i)$, 然后按该主题的**词分布** $b_k \sim \text{Dir}$ 生成词。LDA = “**多主题混合的文档**”。
- **几何直觉**：
把高维 (词表 V) 的文档计数投影到低维 (主题数 K) 的**单纯形**上；单纯形表示**占比**，天然适合**一词多义**的消歧。
- **评价 (困惑度)**：
困惑度 = **逆预测概率几何均值**；越低越好。LDA 通常优于 pLSI、简单混合等 unigram 模型。
- **推断方法**：
 - **折叠 Gibbs**：积分掉 π, B 只采样 q , 更新规则= (词在该主题常见) \times (文档常用该主题)。
 - **变分 (EM/VB)**：有序列版与计数版；VB 还能学习参数后验、鼓励稀疏。
 - **在线 VI**：小批量+递减步长，适合大语料，快且效果接近离线。
- **选主题数 K** ：
用验证、近似证据、变分下界，或用**非参数贝叶斯**让模型自适应。

记号提示：原文在 27.3.4 的若干式子里把 y_{il} 记作 x_{il} 、把 q_{il} 记作 z_{il} ，属排版不统一；按 LDA 语境应分别为词和主题指派。

这一节主要在讲什么？我不太懂

已思考 9s

没事~这小节其实就是把 LDA 主题模型从“是什么、为什么、怎么用、怎么评”讲了个全流程。

一句话版

LDA 把每篇文档看成多个主题按比例混合而成（不是只属于一个主题），并用贝叶斯方法学出：每个主题喜欢哪些词、每篇文档各用多少主题、每个词属于哪个主题。

核心点（超简明）

- 模型设定（生成故事）

1. 给每个文档抽主题配比 $\pi_i \sim \text{Dir}$ (各主题所占比例)；
2. 对文档里每个词位置抽主题 $q_{il} \sim \text{Cat}(\pi_i)$ ；
3. 再按该主题的词分布 $b_k \sim \text{Dir}$ 抽具体词 y_{il} 。

产出：主题词表 $B = \{b_k\}$ 、文档的主题配比 π_i 、每个词的主题标签 q_{il} 。

- 为什么有用

- 允许一篇文档多主题混合（admixture/mixed-membership），比“每文档一个主题”的混合模型灵活。
- 能处理一词多义：同一词可在不同主题中出现（如 *bank*→“银行”或“河岸”）。

- 几何直觉

把高维的“词频向量”（在 V 维单纯形上）投到低维的主题单纯形 ($K < V$)，得到可解释的低维表示（主题占比）。

- 怎么学参数/推断

- 折叠 Gibbs 采样：把 π, B 积分掉，只采每个词的主题 q_{il} ；更新规则≈“该词在该主题常见 × 该文档常用该主题”。
- 变分推断（EM/VB）：用均值场近似；有序列版（对每个词存分布）和计数版（更省内存）。
- 在线变分：小批量 + 递减步长，适合大语料，快且效果接近离线。

- 如何评估好坏

- 用困惑度（perplexity）：平均预测难度的几何均值，越低越好。LDA 通常优于 pLSI、简单混合等 unigram 模型。

- 如何选主题数 K

交叉验证 / 近似证据（AIS、变分下界）/ 非参贝叶斯（如 HDP-LDA）。

记忆小抄

- LDA = 文档的主题配比 π + 主题的词分布 B + 词级主题指派 q 。
- 优点：多主题、可解释、能消歧；常用推断：折叠 Gibbs / 变分（支持在线）。
- 评估用困惑度；选 K 用验证或非参方法。

如果你愿意，我可以用一个超小玩具语料跑一遍折叠 Gibbs 或变分的几步迭代，让你直观看到“主题”和“配比”是怎么长出来的。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

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-occur. The source of the problem is the use of a Dirichlet prior for π_i . The problem with the Dirichlet is 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

$$bk|\gamma \sim \text{Dir}(\gamma V) \quad (27.50)$$

$$zi \sim N(\mu, \Sigma) \quad (27.51)$$

$$\pi_i|zi = S(zi) \quad (27.52)$$

$$qil|\pi_i \sim \text{Cat}(\pi_i) \quad (27.53)$$

$$yil|qil = k, B \sim \text{Cat}(bk) \quad (27.54)$$

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 qil and π_i . Then in the CTM we have

$$yil \sim \text{Cat}(BS(zi)) \quad (27.55)$$

where B is a stochastic matrix. By contrast, in catPCA we have

$y_{il} \sim \text{Cat}(S(W_{zi}))$ (27.56)

where W is an unconstrained matrix.

Fitting this model is tricky, since the prior for π_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 q_{il} 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 Σ^* to a sparse precision matrix $\Sigma^* - 1$ 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.

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:

$b_{t,k}|b_{t-1,k} \sim N(b_{t-1,k}, \sigma^2 V)$ (27.57)

$\pi_{t,i} \sim \text{Dir}(\alpha_1 K)$ (27.58)

$q_{t,il}|\pi_{t,i} \sim \text{Cat}(\pi_{t,i})$ (27.59)

$y_{t,il}|q_{t,il} = k, B_t \sim \text{Cat}(S(b_{t,k}))$ (27.60)

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

One can perform approximate inference 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 $b_{t,k}$ 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. 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 $z_{i1} \in \{0, \dots, C\}$. In addition, each document

has an LDA model associated with it. If $zil = 0$, we generate word yil from the semantic LDA model, with topic specified by qil ; otherwise we generate word yil from the syntactic HMM model. The DGM is shown in Figure 27.13.

The CPDs are as follows:

$$p(\pi_i) = \text{Dir}(\pi_i | \alpha_1 K) \quad (27.61)$$

$$p(q_{il} = k | \pi_i) = \pi_{ik} \quad (27.62)$$

$$p(z_{il} = c | z_{i,l-1} = c) = \text{AHMM}(c, c) \quad (27.63)$$

$$p(y_{il} = v | q_{il} = k, z_{il} = c) = \text{BLDA}(k, v) \text{ if } c = 0 \\ \text{BHMM}(c, v) \text{ if } c > 0 \quad (27.64)$$

where BLDA is the usual topic-word matrix, BHMM is the state-word HMM emission matrix and AHMM 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 corpora4 are shown in Table 27.1. We see that the HMM generally is 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. 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 syntactic 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 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 $y_{il} \in \{1, \dots, V\}$ as usual, but we also have a class label $c_i \in \{1, \dots, C\}$. How can we predict c_i from y_{il} ? 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 $p(c_i, y_{il} | \theta)$. (Blei and McAuliffe 2010) proposes an approach, called supervised LDA, where the class label c_i is generated from the topics as follows:

$$p(c_i | q_i) = \text{Ber}(\text{sigm}(w^T q_i)) \quad (27.65)$$

Here q_i is the empirical topic distribution for document i :
 $q_{ik} = \frac{1}{L_i} \sum_{l=1}^{L_i} q_{ilk} \quad (27.66)$

See Figure 27.15(a) for an illustration.

We can fit this model using Monte Carlo EM: run the collapsed Gibbs sampler in the E step, to compute $E[q_{ik}]$, 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 $p(y_{il} | c_i, \theta)$. The only change from regular LDA is that the topic prior becomes input dependent, as follows:

$$p(q_{il} | \pi_i, c_i = c, \theta) = \text{Cat}(A_c \pi) \quad (27.67)$$

where 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 $x_i \in \mathbb{R}^D$. 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 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 π_i with a deterministic function of the input:

$$\pi_i = S(Wx_i) \quad (27.68)$$

In (Law et al. 2010), this is called multinomial regression LDA. See Figure 27.16(a). Eliminating the deterministic π_i we have

$$p(q_{il}|x_i, W) = \text{Cat}(S(Wx_i)) \quad (27.69)$$

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 y_i ; then treat the inferred π_i as data, and fit a multinomial logistic regression model mapping x_i to π_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 π_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,

$$p(y_{il}|x_i, \theta) = \prod_{l=1}^L p(y_{il}|x_i, \theta) = \prod_{l=1}^L \sum_{k=1}^K p(y_{il}|q_{il}=k, \theta) p(q_{il}=k|x_i, W) \quad (27.70)$$

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 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 W . If $x_i = 1$, then $p(q_{il}|x_i, w_i) = \text{Cat}(S(w_i))$, 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 α a function of the input

$$\alpha_i = \exp(Wx_i) \quad (27.71)$$

where W is a $K \times D$ matrix. Eliminating the deterministic α_i we have

$$\pi_i \sim \text{Dir}(\exp(Wx_i)) \quad (27.72)$$

Unlike (Law et al. 2010), this model allows information to flow between tags via the latent π_i .

A variant of this model, where x_i corresponds to a bag of discrete labels and $\pi_i \sim \text{Dir}(\alpha \cdot x_i)$, is known as labeled LDA (Ramage et al. 2009). In this case, the labels 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:

$$p(z_i|x_i, V) = N(Vx_i, \Sigma) \quad (27.73)$$

$$p(y_i|z_i, W) = \text{Cat}(y_i|S(Wz_i)) \quad (27.74)$$

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 key difference from a neural net is that information can flow between the y_i 's via the latent bottleneck layer 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 x_i to y_i arc, but this would require too many parameters if the number of labels is large.

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 z_i is a mixture of Gaussians using inputdependent 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”.

已思考 26s

下面把整段内容逐句完整翻译，并在每一小节后用通俗、详尽的解释帮你吃透（尽量不依赖数学基础）。最后附上要点总结。原文中的个别拼写/排版小错误（如 *Dirichelt* 应为 *Dirichlet*、*co-occcur* 应为 *co-occur*、*infernece* 应为 *inference* 等）保持忠实翻译，并在解释里指出不影响理解。

27.4 LDA 的扩展

自 2003 年第一篇论文发表以来，人们提出了许多 LDA 的扩展。下面我们简要讨论其中的一些。

27.4.1 相关主题模型 (Correlated topic model, CTM)

LDA 的一个弱点是无法捕捉主题之间的相关性。例如，如果一篇文档有“商业 (business) ”主题，那么合理地期望“金融 (finance) ”主题也会共同出现。问题的根源在于对 π_i 使用了 Dirichlet 先验。Dirichlet 的问题在于它只由一个均值向量和一个强度参数刻画，但它的协方差是固定的 ($\Sigma_{ij} = -\alpha_i \alpha_j$)，而不是一个可自由设定的参数。

一种变通方法是用**逻辑正态分布 (logistic normal) **取代 Dirichlet 先验，类似第 27.2.2 节的分类 PCA (categorical PCA)。模型变为

$$b_k \mid \gamma \sim \text{Dir}(\gamma \mathbf{1}_V) \quad (27.50)$$

$$z_i \sim N(\mu, \Sigma) \quad (27.51)$$

$$\pi_i \mid z_i = S(z_i) \quad (27.52)$$

$$q_{il} \mid \pi_i \sim \text{Cat}(\pi_i) \quad (27.53)$$

$$y_{il} \mid q_{il} = k, B \sim \text{Cat}(b_k) \quad (27.54)$$

这称为**相关主题模型** (CTM; Blei and Lafferty, 2007)。这和分类 PCA 非常相似，但略有不同。为看清差别，让我们把 q_{il} 与 π_i 边缘化。在 CTM 中，我们有

$$y_{il} \sim \text{Cat}(B S(z_i)) \quad (27.55)$$

其中 B 是一个**随机矩阵**（每列为一个主题的词分布，列和为 1）。相比之下，在分类 PCA (catPCA) 中有

$$y_{il} \sim \text{Cat}(S(W z_i)) \quad (27.56)$$

其中 W 是一个**不受约束**的矩阵。

拟合这个模型比较棘手，因为此时 π_i 的先验不再与 q_{il} 的多项分布似然共轭。不过我们可以使用第 21.8.1.1 节讨论的任何变分方法（那里讲了贝叶斯多类逻辑回归）。在 CTM 中更难一些，因为类别型响应变量 q_i 是隐藏的，但可以通过再引入一个均值场近似来处理。细节见 (Blei and Lafferty, 2007)。

拟合好模型后，可以把估计得到的 $\hat{\Sigma}$ 通过剪枝弱边转成一个稀疏精度矩阵 $\hat{\Sigma}^{-1}$ ，得到一个稀疏高斯图模型。这有助于可视化主题之间的相关性。图 27.10 展示了将该过程应用到 1990–1999 年的 *Science* 杂志文章的结果。（该语料在去停用词和去低频词后包含 16,351 篇文档与 570 万个词（其中 19,088 个不同词）。）节点表示主题，节点内列出该主题的前 5 个代表词。字体大小反映主题在语料中的整体流行度。边表示精度矩阵中的显著元素。

通俗解释

- LDA 默认假设“主题彼此独立”，所以学不出“商业→金融”这种共现关系。
 - CTM 的招数：先用高斯在隐空间建模相关性（可用任意协方差 Σ ），再用 softmax 变成主题占比 π_i 。这样主题间的相关结构就能学到。
 - 和 catPCA 的区别：CTM 先用 $S(z)$ 得到主题占比，再乘以主题-词矩阵 B 得词概率；catPCA 直接把 z 线性变换后 softmax 成词概率。
 - 实用上，CTM 拟合更难（不共轭），通常用变分近似。最后把 $\hat{\Sigma}^{-1}$ 画成图，就能看到“哪些主题互相关联”。
-

27.4.2 动态主题模型 (Dynamic topic model, DTM)

在 LDA 中，主题（词分布）被假设为静态。有时让这些分布随时间平滑演化更合理。例如，一篇文章可能使用“神经科学”主题；但如果写于 1900s，更可能用“nerve”这样的词，而写于 2000s 则更可能用“calcium receptor”（反映了神经科学向分子生物学发展的总体趋势）。

一种建模方法是使用动态逻辑正态模型，如图 27.11 所示。具体来说，我们假设主题分布按高斯随机游走演化，然后将这些高斯向量通过 softmax 映射为概率：

$$b_{t,k} \mid b_{t-1,k} \sim N(b_{t-1,k}, \sigma^2 I_V) \quad (27.57)$$

$$\pi_i^t \sim Dir(\alpha \mathbf{1}_K) \quad (27.58)$$

$$q_{il}^t \mid \pi_i^t \sim Cat(\pi_i^t) \quad (27.59)$$

$$y_{il}^t \mid q_{il}^t = k, B_t \sim Cat(S(b_{t,k})) \quad (27.60)$$

这称为动态主题模型 (DTM; Blei and Lafferty, 2006b)。

在该模型中，可以用**结构化均值场**（第 21.4 节）做近似推断，利用 Kalman 平滑（第 18.3.1 节）在 $b_{t,k}$ 这些**线性高斯链**上做**精确推断**（细节见 Blei and Lafferty, 2006b）。

图 27.12 展示了将该系统应用到 *Science* 杂志 100 年文章时的典型输出。上方：每隔 10 年可视化某一主题（似乎与神经科学相关）的**前 10 个词**；左下：画出若干特定词**属于该主题的概率**随时间的变化；右下：列出一些包含该主题的文章标题。

一个有趣应用是**时间校正的文档检索**。例如，搜索“遗传病的遗传机制”相关文档，现代论文会用“DNA”，但更早（发现 DNA 之前）的论文可能用“heritable unit”等术语；然而它们可能是**同一主题**。类似思想也可用于**跨语言检索**（如 Cimiano et al., 2009）。

通俗解释

- DTM 让“主题的关键词偏好”随时间**缓慢漂移**，更贴合真实语义演化（新术语出现、旧术语退场）。
 - 算法上，时间维度上是**高斯状态空间模型**（可用 Kalman 平滑），词生成仍像 LDA。
 - 好处：能做“按时代对齐”的检索、主题演化分析（看到“神经科学”从“神经/神经元”走向“受体/钙通道”等）。
-

27.4.3 LDA-HMM

LDA 假设词是可交换的（不考虑顺序），这显然不符合事实。建模词之间的**顺序依赖**的一个简单方法是使用**隐马尔可夫模型（HMM）**。但 HMM 的问题是它只能建模**短程依赖**，抓不住文档的整体主旨；因此它能生成**句法上正确的句子**（见表 17.1），却未必**语义通顺**。

可以把 LDA 与 HMM 结合，得到 LDA-HMM (Griffiths et al., 2004)。这个模型让 HMM 的**状态去生成功能/句法词**（如 “and”“however”），而让 LDA 来生成**内容/语义词**（更难预测）。模型中有一个**特殊的 HMM 状态**规定“此处由 LDA 生成词”；其它时候由 HMM 生成。

更形式化地，对每篇文档 i ，模型定义了一个 HMM，状态 $z_{il} \in \{0, \dots, C\}$ 。此外，每篇文档有关联一个 LDA 模型。若 $z_{il} = 0$ ，则从**语义 LDA 模型**（主题由 q_{il} 指定）生成词 y_{il} ；否则，从**句法 HMM 模型**生成 y_{il} 。其有向图模型见图 27.13。条件概率如下：

$$p(\pi_i) = \text{Dir}(\pi_i \mid \alpha \mathbf{1}_K) \quad (27.61)$$

$$p(q_{il} = k \mid \pi_i) = \pi_{ik} \quad (27.62)$$

$$p(z_{il} = c' \mid z_{i,l-1} = c) = A_{\text{HMM}}(c, c') \quad (27.63)$$

$$p(y_{il} = v \mid q_{il} = k, z_{il} = c) = \begin{cases} B_{\text{LDA}}(k, v), & c = 0 \\ B_{\text{HMM}}(c, v), & c > 0 \end{cases} \quad (27.64)$$

其中 B_{LDA} 是常规的主题-词矩阵， B_{HMM} 是 HMM 的**状态-词**发射矩阵， A_{HMM} 是 HMM 的**状态转移**矩阵。

该模型中的推断可以用**折叠 Gibbs 采样**完成，即**解析积分**掉所有连续量。细节见 Griffiths et al. (2004)。

将该模型（设置 $K = 200$ 个 LDA 主题、 $C = 20$ 个 HMM 状态）应用到 Brown+TASA 语料的结果见表 27.1。可以看到，HMM 通常负责**句法词**，而 LDA 负责**语义词**。若没有 HMM，LDA 的主题会被功能词“污染”（见图上方），这也是预处理时常常去除此类词的原因。

该模型还能帮助**消歧**：同一个词是**句法用法**还是**语义用法**。图 27.14 给出在 NIPS 语料上的例子：比如 “we require the algorithm to return a matrix”（动词） vs “the maximal expected return”（名词）。理论上词性标注器也能区分，但请注意：（1）LDA-HMM 是**完全无监督**（未用词性标注）；（2）有时词性相同但**语义不同**，如 “the left graph”（句法修饰） vs “the graph G”（语义名词）。

关于**语法与语义**的概率模型是一个非常庞大的主题，这里不展开。更多信息见 Jurafsky and Martin (2008) 等。

通俗解释

- 直觉：**HMM 管语法骨架**（连词、介词、结构词），**LDA 管语义血肉**（内容词）。
 - 好处：主题不再被“and / the / of ...”这类词干扰；还能区分一个词在不同上下文中的**角色**（语法 vs 语义）。
-

27.4.4 监督式 LDA (Supervised LDA)

本节讨论把 LDA 扩展到**不仅仅使用词**、还能利用各种**侧信息**的情形。

27.4.4.1 生成式监督 LDA

假设我们仍有一个可变长度的词序列 $y_{il} \in \{1, \dots, V\}$ ，同时还拥有一个**类别标签** $c_i \in \{1, \dots, C\}$ 。怎样根据 y_i 预测 c_i 呢？有许多方法，但大多是从词到类的**直接映射**。某些任务（如情感分析）中，先进行**推断**来尽量**消除词义歧义**会更好。例如，我们要判断一篇影评是否正面。遇到 “Brad Pitt was excellent until the middle of the movie”，词 “excellent” 可能让我们误判为正面，但显然整体情感是**负面**。

解决这类问题的一种方法是构建**联合模型** $p(c_i, y_i \mid \theta)$ 。 (Blei and McAuliffe, 2010) 提出了**监督 LDA (sLDA)**：令类别标签 c_i 由主题生成，如下

$$p(c_i \mid q_i) = \text{Ber}(\text{sigm}(w^\top q_i)) \quad (27.65)$$

这里 q_i 是文档 i 的**经验主题分布**：

$$q_{ik} = \frac{1}{L_i} \sum_{l=1}^{L_i} q_{ilk} \quad (27.66)$$

示意见图 27.15(a)。

可用 Monte Carlo EM 拟合该模型：E 步运行**折叠 Gibbs**，计算 $E[q_{ik}]$ ；然后把它当作输入特征，丢给一个标准的**逻辑回归**来更新 w 。

通俗解释

- sLDA 的思路：先用主题模型把“语义结构”捞出来，再用这份结构去预测标签。这样能**减小误导词**（如 “excellent”）的影响。
 - 公式 (27.65) 写的是**二分类**（伯努利 + sigmoid）；更一般可换成其他 GLM 链接函数。
-

27.4.4.2 判别式监督 LDA

另一种方法称为**判别式 LDA** (Lacoste-Julien et al., 2009)，见图 27.15(b)。这是一个**判别式模型** $p(y_i \mid c_i, \theta)$ 。与常规 LDA 唯一的不同，是**主题先验变为依赖输入**：

$$p(q_{il} \mid \pi_i, c_i = c, \theta) = \text{Cat}(A_c \pi) \quad (27.67)$$

其中 A_c 是一个 $K \times K$ 的**随机矩阵**。

到目前为止，我们假设“侧信息”是一个**单一类别变量** c_i 。但实际中常有**高维协变量** $x_i \in \mathbb{R}^D$ 。例如图像标注任务： y_{il} 表示**相关标签/标记**，目标是给定 x_i 预测它们。下面讨论若干把 LDA 扩展到“给定输入生成标签”的尝试。

最简单的方法是用多输出的专家混合 (§11.2.4)。这和 LDA 很像，只是把 π_i 的 Dirichlet 先验换成输入的**确定性函数**：

$$\pi_i = S(Wx_i) \quad (27.68)$$

在 (Law et al., 2010) 中称为**多项回归 LDA**。见图 27.16(a)。消去确定性的 π_i ，得到

$$p(q_{il} \mid x_i, W) = \text{Cat}(S(Wx_i)) \quad (27.69)$$

可按常规用 EM 拟合。不过 (Law et al., 2010) 建议另一做法：先只基于 y_i 拟合**无监督 LDA**；再把推断到的 π_i 当数据，用**多项逻辑回归**拟合 $x_i \mapsto \pi_i$ 。尽管很快，但“先无监督 LDA 再回归”的潜在变量未必**判别性强**（见 Blei and McAuliffe, 2010）。

这个模型还有一个更微妙的问题：因为 π_i 是输入的**确定性函数**，它等价于**被观测到**，从而 q_{il} （进而标签 y_{il} ）之间**相互独立**。换言之，

$$p(y_i | x_i, \theta) = \prod_{l=1}^{L_i} p(y_{il} | x_i, \theta) = \prod_{l=1}^{L_i} \sum_k p(y_{il} | q_{il} = k, B) p(q_{il} = k | x_i, W) \quad (27.70)$$

这意味着如果我们观测到了某个标签的取值，它对**其他标签**没有任何影响。这可能解释了 (Law et al., 2010) 的结果为何只比**独立预测各标签**略有提升。

引入相关性的一种办法是把 W 设为**随机变量**。模型如图 27.16(b)，称为**随机效应专家混合**。通常给 W_i （或文档特定权重）一个**高斯先验**。当 $x_i = 1$ （无输入）时， $p(q_{il} | x_i, w_i) = \text{Cat}(S(w_i))$ ，就**回到 CTM**。还可以给 q_{il} 加上**马尔可夫动态**，叫**条件主题随机场** (Zhu and Xing, 2010)。

一个密切相关的方法称为**Dirichlet 多项回归 LDA** (Dirichlet Multinomial Regression LDA; Mimno and McCallum, 2008)，见图 27.16(c)。它与标准 LDA 相同，但把 α 设为输入的函数：

$$\alpha_i = \exp(Wx_i) \quad (27.71)$$

其中 W 是 $K \times D$ 矩阵。消去确定性的 α_i ，我们有

$$\pi_i \sim \text{Dir}(\exp(Wx_i)) \quad (27.72)$$

与 (Law et al., 2010) 不同，该模型允许信息通过**潜在的** π_i 在标签之间**相互流动**。

该模型的一个变体是**labeled LDA** (Ramage et al., 2009)，其中 x_i 对应一个**离散标签的词袋**，且 $\pi_i \sim \text{Dir}(\alpha^{\$}x_i)$ （原文记号 $\alpha^{\$}x_i$ 为排版写法，可理解为“由标签选择出的超参数向量”）。在这种情况下， x_i 中的**每个标签与一个潜在主题一一对应**，使得学习到的主题**更易解释**。其扩展 **partially labeled LDA** (Ramage et al., 2011) 允许**每个标签拥有多个潜在子主题**；该模型将 LDA / labeled LDA / 多项混合模型 作为**特例统一起来**。

通俗解释

- “**判别式** vs “**生成式**”：生成式是“先解释语义→再生成标签/类”；判别式是“给定类/输入→生成词/标签”。
- **输入驱动的主题先验**：让输入 x 或类 c 改变“这篇文档更可能用哪些主题”。

- **大坑：**若把 π_i 设成 x_i 的**确定性函数**，标签之间就**断开了联系**（彼此独立），学不出“**标签-标签相关性**”。
 - **解法：**给参数 W 加**随机性**（随机效应），或让 α_i 依赖 x_i （仍保持 π_i 随机），从而**恢复标签间的相互影响**。
 - **labeled LDA：**把外部标签直接与主题**绑定**，主题更好解释；部分标注版允许“一个标签下的多子主题”。
-

27.4.4.3 判别式分类 PCA (Discriminative categorical PCA)

不用 LDA 的另一条路，是把**分类 PCA (catPCA) 加上输入**，见图 27.17(a)。由于潜在空间现在是**实数向量**，我们可以用**线性回归**做“输入→隐层”的映射；“隐层→输出”则用传统 catPCA：

$$p(z_i | x_i, V) = N(Vx_i, \Sigma) \quad (27.73)$$

$$p(y_i | z_i, W) = \prod_l \text{Cat}(y_{il} | S(Wz_i)) \quad (27.74)$$

该模型本质上是一个**一隐层的概率神经网络**，见图 27.17(b)，但**输出是可交换的**（可处理“**标签数可变**”的情况）。与普通神经网不同之处在于：信息可以通过**隐层瓶颈** z_i 在各个 y_{il} 之间**流动**。当在给定特征后**输出标签之间仍强相关**时（多标签分类常见），这应当比常规神经网更好。注意我们也可加一条直接的 $x_i \rightarrow y_i$ 边，但在标签很多时，这会需要**过多参数**。

可用对第 12.4 节**变分 EM**作小改动来拟合该模型。若把该模型用于**回归而非分类**，则可通过修改**因子分析**的 EM，在 E 步精确求解。（Ma et al., 1997）报告该方法**收敛比标准反向传播更快**。

还可扩展：让 z_i 的先验变为**输入相关均值的高斯混合**。若输出是高斯，就得到**判别式因子分析混合**（Fokoue, 2005；Zhou and Liu, 2008）。若输出是类别型，将得到一个（尚未发表的）模型，可称为“**判别式分类因子分析混合**”。

通俗解释

- 这是“**带输入的 catPCA**”：
 - $x \rightarrow z$: 线性高斯（像回归到隐向量）。
 - $z \rightarrow y$: softmax 多项 (catPCA)。
- **关键优势：**隐层 z 让标签之间**“看见彼此”（通过共享的低维表示相关联），更适合多标签**任务。
- **训练：**变分 EM；做回归时还能把 E 步做成**闭式**（快）。

- **再升级：**让 z 的先验是“输入驱动的多个簇”的混合，可看作“多专家 + 因子分析”的判别式版本；对类别输出是一个自然但尚未发表的扩展。

要点小结（记忆版）

- **CTM（相关主题）：**用高斯→softmax取代 Dirichlet，以显式建模主题相关性；训练靠变分近似；可把 $\hat{\Sigma}^{-1}$ 画成主题关联图。
- **DTM（动态主题）：**主题词分布随时间按高斯随机游走演化（再 softmax）；用 Kalman 平滑 + 结构化均值场推断；能做时间对齐检索、看主题演化。
- **LDA-HMM：**HMM 抓句法序列，LDA 抓语义内容；功能词交给 HMM，避免污染主题；还能无监督地区分词的语法/语义用法。
- **监督 LDA (sLDA)：**在 LDA 上为标签建立生成关系（如逻辑回归链接），先推断主题再预测标签，能缓解误导词。
- **判别式 LDA 与输入驱动扩展：**
 - 把先验 π 或 α 设为输入的函数，但切忌把 π 变成完全确定，否则标签彼此独立。
 - 解决：给参数加随机效应或让 $\alpha(x)$ 而 π 仍随机（如 DMR-LDA），从而保留标签间相关性。
 - labeled / partially labeled LDA：把外部标签与主题绑定（可扩展为多子主题），提升可解释性。
- **判别式 catPCA：**用实值隐层承接输入与多标签输出，输出间通过瓶颈互相影响；变分 EM 拟合，可扩展成输入依赖的混合版本。

小提示：正文里关于 Dirichlet 协方差的写法是“固定 $\Sigma_{ij} = -\alpha_i \alpha_j$ ”的直观表达，严格形式有缩放项，但核心观点是不允许任意协方差结构；CTM 正是为此改用逻辑正态来放宽相关性。

这一节主要在讲什么？我不太懂

没关系！这段（27.4）其实是在说：LDA“基础款”很好用，但有不少缺点；后来大家做了很多“升级包”来补齐它的短板。主要升级思路有 5 类——各自解决一个痛点。

一句话总览

把 LDA 想成“配方生成器”（主题=配方，文档=多配方混合）。27.4 讲的是：如何让配方彼此有关联、会随时间变化、能考虑词序、能用上标签/特征、以及用别的潜在表示法来做多

标签。

5 类扩展在解决什么？

1. CTM：相关主题模型

- 痛点：LDA 假设主题彼此独立，但现实里“商业→金融”常一起出现。
- 做法：用高斯→softmax代替 Dirichlet，让**主题之间能相关**；学完还能画“主题关系图”。

2. DTM：动态主题模型

- 痛点：话题会**随时间演化**（老论文说“nerve”，新论文说“calcium receptor”）。
- 做法：让每个主题的词分布**随时间平滑漂移**（高斯随机游走 + softmax）；可做**时间对齐检索与“主题演化”可视化**。

3. LDA-HMM：把语义和句法拆着建

- 痛点：LDA不看**词序**，HMM只抓**短程**且容易被语义拖后腿。
- 做法：**HMM 管句法词**（and, however...），**LDA 管内容词**；能减少功能词污染，还能**无监督**区分同一词的“语法/语义”用法。

4. 监督式 LDA（带标签/输入的变体）

- 目标：不只“发现主题”，还要用**主题去预测标签**，或让**输入影响主题**。
- 代表：
 - **sLDA (生成式)**：先推断文档的“主题配比”，再用它去**生成/预测标签**（如情感）。
 - **判别式 LDA**：给定类/输入来**生成词**（把先验做成依赖输入/类别）。
 - **输入驱动的 LDA**：让 α 或 π 受**输入控制**（如 DMR-LDA、labeled LDA/partially labeled LDA）。
- 大坑：**不要把 π 设成输入的完全确定函数**，否则各标签彼此独立、学不出标签间相关性；解法是让 π 仍然是**随机的**（或给参数加随机效应）。

5. 判别式 categorical PCA (catPCA+输入)

- 痛点：多标签任务里，**标签彼此强相关**，普通神经网难共享信息。
- 做法：用**实值隐层 z** 做“瓶颈”，输入线性回归到 z ，再用 catPCA (softmax) 生成多标签；标签间通过 z **互相“看见”**，适合多标签。

什么时候用哪一个？（小抄）

- 想看**主题之间怎么联动** → CTM。
- 想看**主题随时间的演变/做年代对齐检索** → DTM。
- 想兼顾**词序（句法）和主题（语义）** → LDA-HMM。

- **有标签或外部特征**, 要预测/解释 → sLDA、判别式/输入驱动的 LDA (优先让 π 仍随机)。
- **多标签强相关且有输入** → 判别式 catPCA (隐层共享)。

要点小结

- 27.4 是 LDA 的**应用与增强版图**: 相关性 (CTM) 、时间 (DTM) 、序列 (LDA-HMM) 、监督/输入 (sLDA 等) 、以及用 catPCA 做多标签。
- 共同主题: **让模型更贴近真实数据的结构** (相关、演化、序列、监督、标签间依赖)。
- 实操提醒: CTM/DTM 常用**变分推断**; 监督/输入模型要避免让 π 变成**确定量**, 否则会切断输出间的联系。

翻译并解释以下内容, 翻译不能有遗漏, 解释请通俗易懂且详尽, 因为我的数学基础不太好, 最后总结要点。

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 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. Further 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, $B1 = \{1, 4, 6\}$, $B2 = \{2, 3, 5, 8\}$, and $B3 = \{7, 9\}$, such that most of the connections go from nodes in $B1$ to $B2$, or from $B2$ to $B3$, or from $B3$ to $B1$. This is illustrated in Figure 27.18(b).

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 permuted 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 $q_i \sim \text{Cat}(\pi)$, where π_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:

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

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. 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 π 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

$$p(R_{ij} = r | q_i = a, q_j = b, x_i, x_j, \theta) = Ber(r | w^T a, b f(x_i, x_j)) \quad (27.76)$$

where $f(x_i, x_j)$ is some way of combining the feature vectors. For example, we could use concatenation, $[x_i, x_j]$, or elementwise product $x_i \otimes x_j$ 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 $q_i \in \{1, \dots, K\}$ with $\pi_i \in S^K$. This is known as the mixed membership stochastic block model, and is similar in spirit to fuzzy clustering or soft clustering. Note that π_{ik} is not the same as $p(z_i = k | D)$; the former represents ontological uncertainty (to what degree does each object belong to a cluster) whereas the latter represents epistemological uncertainty (which cluster does an object belong to). If we want to combine epistemological and ontological uncertainty, we can compute $p(\pi_i | D)$.

In more detail, the generative process is as follows. First, each node picks a distribution over blocks, $\pi_i \sim Dir(\alpha)$. 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:

$$q_{i \rightarrow j} \sim Cat(\pi_i), q_{i \leftarrow j} \sim Cat(\pi_j) \quad (27.77)$$

Finally, generate each edge R_{ij} using the following model:
 $p(R_{ij} = 1 | q_{i \rightarrow j} = a, q_{i \leftarrow j} = b, \eta) = \eta_{a,b} \quad (27.78)$

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 $E[\pi]_i$ 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 ρ , 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:

$$p(R_{ij} = r | q_i \rightarrow j = a, q_j \leftarrow j = b, \eta) = \rho \delta_0(r) + (1 - \rho) \text{Ber}(r | \eta a, b) \quad (27.79)$$

See (Airoldi et al. 2008) for details.

27.5.3 Relational topic model

In many cases, the nodes in our network have attributes. 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 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:

$$p(R_{ij} = 1 | q_i, q_j, \theta) = \text{sigm}(w^T(q_i \otimes q_j) + w_0) \quad (27.80)$$

Recall that q_i is the empirical topic distribution for document i , $q_{ik} = 1 / \sum_{l=1}^L q_{il}$. See Figure 27.23

Note that it is important that R_{ij} depend on the actual topics chosen, q_i and q_j , and not on the topic distributions, π_i and π_j , otherwise predictive performance is not as good. The intuitive reason for this is as follows: if

R_{ij} is a child of π_i and π_j , it will be treated as just another word, similar to the q_i 's and y_i '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 q_i and 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 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.

下面把整段原文逐句完整翻译，并在每个小节后给出**通俗、详尽的解释**（尽量少用数学）。最后附上**要点小结**。原文里个别拼写小错误（如 *permuted* 应为 *permuted*, *Futher* 应为 *Further*）不影响理解。

27.5 面向图结构数据的潜在变量模型 (LVMs for graph-structured data)

原文翻译

另一类离散数据来源于对图 (graph) 或网络结构的建模。为了看清二者的联系，回忆一下：任意一个有 D 个节点的图，都可以表示为一个 $D \times D$ 的邻接矩阵 G ，其中当且仅当从节点 i 指向节点 j 有一条边时， $G(i,j) = 1$ 。这类矩阵是**二值的**，且往往**非常稀疏**。参见图 27.19 的示例。

图在许多应用领域中都会出现，比如**社交网络建模**、**蛋白—蛋白相互作用网络**，或者**在人或动物之间疾病传播的模式**。分析这类数据通常有两个主要目标：第一，尝试在图中发现某种“**有趣的结构**”，例如**簇**或**社群**；第二，尝试**预测**未来可能出现的**连边**（例如“谁将与谁成为朋友”）。下面我们概述一些被提出用来完成这些任务的模型，其中一些与 LDA 有关。关于这些方法和其他方法的更多细节，见 (Goldenberg et al., 2009) 及其中的参考文献。

27.5.1 随机分块模型 (Stochastic block model, SBM)

原文翻译

在图 27.18(a) 中，我们展示了一个有 9 个节点的**有向图**。乍看之下没有明显结构。然而，如果深入观察，我们会发现可以把节点划分为 3 个组或**块**： $B_1 = \{1, 4, 6\}$ 、 $B_2 = \{2, 3, 5, 8\}$ 、 $B_3 = \{7, 9\}$ ，并且**大多数连接**是从 B_1 的节点连到 B_2 ，或从 B_2 连到 B_3 ，或从 B_3 连到 B_1 。这在图 27.18(b) 中做了说明。

如果我们画出邻接矩阵，这个问题更容易理解。图 27.19(a) 给出了按**原始节点顺序**排列的矩阵；图 27.19(b) 给出了按**置换后顺序 (permuted)** 排列的矩阵。显然可以看到块结构。

我们可以这样为具有分块结构的图建立一个**生成模型**。第一，对每个节点，抽取一个**潜在块** $q_i \sim \text{Cat}(\pi)$ ，其中 π_k 是选择第 k 个块的概率 ($k = 1:K$)。第二，为所有块对 (a, b) 选择“从组 a 连向组 b ”的**连边概率** $\eta_{a,b}$ ；这可以来自一个 **Beta 先验**。最后，用下面的模型生成每条边 R_{ij} ：

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

这称为**随机分块模型** (Nowicki and Snijders, 2001)。图 27.21(a) 用 DGM (有向图模型) 展示了该模型；图 27.19(c) 展示了如何用该模型对示例中的节点进行**聚类**。

注意这与常规聚类问题有很大不同。比如，我们看到块 3 中所有节点被分到一起，**即使它们之间没有任何边**。它们共享的性质是：它们“倾向于”连到块 1 的节点，且“倾向于”从块 2 的节点接收连边。图 27.20 展示了该模型在生成多种不同图结构时的能力。例如，一些社交网络具有**层级结构**，可以通过把人按不同**社会层次**聚类来建模；而另一些由一组**小团体 (cliques) **构成。

不同于标准混合模型，这个模型**无法**用精确 EM 来拟合，因为所有潜在的 q_i 变量会变得**相关**。不过，可以使用**变分 EM** (Airoldi et al., 2008)、**折叠 Gibbs 采样** (Kemp et al., 2006) 等方法。我们略去细节（与 LDA 的情况相似）。

在 (Kemp et al., 2006) 中，他们放宽了“块数 K 固定”的限制：用**狄利克雷过程**（见 §25.2.2）替换 π 上的 Dirichlet 先验。这被称为**无限关系模型**。详见 §27.6.1。

如果每个节点都有**特征**，我们可以把该模型做成一个**判别式版本**，例如定义

$$p(R_{ij} = r \mid q_i = a, q_j = b, x_i, x_j, \theta) = \text{Ber}(r \mid w_{a,b}^\top f(x_i, x_j)) \quad (27.76)$$

其中 $f(x_i, x_j)$ 是把两个特征向量组合起来的一种方式。例如，我们可以使用**拼接** $[x_i, x_j]$ ，或**逐元素乘积** $x_i \otimes x_j$ （如监督式 LDA 中那样）。**整体模型**类似于**专家混合模型**的**关系扩展**。

通俗解释

- **SBM 的直觉**: 给每个节点一个**组别标签**(块)，再让“**块对**(a, b)之间的连边概率”控制谁更可能连谁。所以**相似性**不是“节点对看起来像不像”，而是“它们对**其他块的连边模式**像不像”。
 - **与普通聚类的区别**: 即便同一块的节点**彼此不相连**，也能被分在一起——因为它们的**对外连接偏好一致**。
 - **拟合**: 像 LDA 一样，潜在变量相互牵连，不能 Exact EM；常用**变分**或Gibbs。
 - **可扩展**: 块数未知时，用**非参数贝叶斯 (DP) **得到“**无限**”版本（自动决定簇数）。
 - **带节点特征**: 可把“是否连边”的概率设为**特征的函数**（逻辑回归形式），并且**按块对**有不同权重 $w_{a,b}$ 。
-

27.5.2 混合成员随机分块模型 (Mixed membership stochastic block model, MMSBM)

原文翻译

在 (Airoldi et al., 2008) 中，他们放宽了“每个节点只属于一个簇”的限制。也就是说，用 $\pi_i \in S_K$ 取代了 $q_i \in \{1, \dots, K\}$ 。这称为**混合成员随机分块模型**，在精神上类似于**模糊聚类/软聚类**。注意 π_{ik} 不等同于 $p(z_i = k | D)$ ：前者表示**本体论不确定性**（一个对象在**多大程度上**属于各个簇），而后者表示**认识论不确定性**（对象**属于哪个簇**）。如果我们想把“认识论+本体论不确定性”结合起来，可以计算 $p(\pi_i | D)$ 。

更详细地，生成过程如下。第一，每个节点抽取一个关于块的分布 $\pi_i \sim \text{Dir}(\alpha)$ 。第二，为所有块对选择“从组 a 连向组 b ”的概率， $\eta_{a,b} \sim \text{Beta}(\alpha, \beta)$ 。第三，对每条边，为**两个方向**分别抽取一个离散变量：

$$q_{i \rightarrow j} \sim \text{Cat}(\pi_i), \quad q_{i \leftarrow j} \sim \text{Cat}(\pi_j) \quad (27.77)$$

最后，用下面的模型生成每条边 R_{ij} ：

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

DGM 见图 27.21(b)。

不同于常规 SBM，在 MMSBM 中，每个节点可以**根据它连接的对象不同而扮演不同角色**。为说明这一点，我们看一个社会网络分析文献中广泛使用的数据集。数据描述了**18位修道士**之间“谁喜欢谁”。它由 Sampson (1968) 在数月内**手工收集**。（如今在 Facebook 之类的社交媒体时代，18 人的网络微不足道，但我们讨论的方法可以扩展到更大规模。）图 27.22(a) 画出了原始数据，图 27.22(b) 画出了每位修道士在 $K = 3$ 情况下的 $E[\pi]_i$ 。我们

看到，大多数修道士属于三个簇之一，称为“**年轻激进派**（young turks）”“**被排斥者**（outcasts）”“**忠诚反对派**（loyal opposition）”。然而一些人，尤其是第 15 号修道士，同时属于两个簇；Sampson 把这些人称为“**摇摆者**（waverers）”。有趣的是，模型能复现 Sampson 当年手工得到的洞见。

社会网络分析中一个普遍问题是**缺失数据**。例如，当 $R_{ij} = 0$ 时，这可能是因为 i 与 j 没有机会互动，或者该互动的数据不可得，而不是他们不愿互动。换句话说，“**缺少证据**”并不等于“**证据缺失**”。我们可以通过修改观测模型来刻画这一点：以概率 ρ 用**背景模型**生成一个 0，而只以概率 $1 - \rho$ 强制模型去解释观测到的 0。也就是说，我们对观测模型做**稳健化**以允许**离群点**，如下：

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

详见 (Airoldi et al., 2008)。

通俗解释

- **MMSBM 的直觉**：每个节点不是“只属于一个社群”，而是有一个**社群占比向量** π_i ；当它跟别人互动时，会从自己的 π_i 里**抽一个“扮演的角色”**（发起方用 $q_{i \rightarrow j}$ ，接收方用 $q_{i \leftarrow j}$ ），再看两角色 (a, b) 的连边概率 $\eta_{a,b}$ 。
- **价值**：能发现“**跨社群的人**”（摇摆者、多重身份），更贴合现实。
- **缺失的 0**：把一部分 0 视为“**未知/没机会**”，不强行解释；只对 $1 - \rho$ 的 0 负责，模型**更稳健**。

27.5.3 关系主题模型 (Relational topic model, RTM)

原文翻译

在很多情况下，网络中的节点带有**属性**。例如，如果节点表示学术论文，边表示**引用**，那么属性就包括**论文正文的文本**。因此，构造一个**能同时解释文本和链接结构**的模型是有益的。这类模型可以在给定文本时预测连边，甚至反过来也是可能的。

关系主题模型 (RTM) (Chang and Blei, 2010) 就是实现这一目标的一种方式。它是**监督式 LDA** (§27.4.4.1) 的一个简单扩展，其中**响应变量** R_{ij} (表示节点 i 与 j 之间是否有边) 建模如下：

$$p(R_{ij} = 1 \mid q_i, q_j, \theta) = \text{sigm}(w^\top (q_i \otimes q_j) + w_0) \quad (27.80)$$

回忆 q_i 是文档 i 的**经验主题分布**， $q_{ik} = \frac{1}{L_i} \sum_{l=1}^{L_i} q_{ilk}$ 。参见图 27.23。

注意，关键在于：令 R_{ij} 依赖于实际选择的主题 q_i, q_j ，而不是依赖主题分布 π_i, π_j 。否则，预测性能不够好。直觉原因是：如果让 R_{ij} 成为 π_i, π_j 的子节点，它会被当成“**又一个词**”，类似于 q_{il} 与 y_{il} ；但**词比边多得多**，图结构信息会被“稀释”。如果让 R_{ij} 成为 q_i, q_j 的子节点，图信息就能**更直接**影响主题的选择。

该模型可以用与 sLDA 类似的方式来拟合。详见 (Chang and Blei, 2010)。这种方法在**预测缺失连边**方面比更简单的两步法（先拟合 LDA，再把 q_i 当逻辑回归的输入）表现更好。其原因类似于**偏最小二乘 (PLS, §12.5.2)** 优于 PCA+线性回归：RTM 学到的潜在空间被迫同时对**图结构与词**有预测力，而 LDA 学到的潜在空间则未必有利于预测图。

通俗解释

- 目标**：一边学“每篇文档的主题”（内容），一边学“论文之间是否引用”（链接）。
 - 做法**：把“是否有边”当成由“两个文档的**主题组合** $q_i \otimes q_j$ ”喂给一个**逻辑回归**来决定。
 - 为什么依赖 q 而不是 π** ： π 太“平均”，把链接当“另一个词”会被海量词信息淹没；用 q （词位级/经验主题）更能让**链接信号**影响主题推断。
 - 端到端训练**：共同学习“文本—主题—链接”这条链，比“先 LDA 后回归”的**后处理**更准。
-

通俗总览 (把 27.5 讲成白话)

- 数据长相**：图 = 邻接矩阵 (0/1、很稀疏)。任务 = **找结构**（社群/层级/角色）+ 补边/预测边。
 - SBM**：每个节点一个**块标签**；边由**块对**的连边率决定。找的是“**连接偏好相似**”的节点群，而不是“彼此相连的”节点群。
 - MMSBM**：节点有多重身份占比 π ；对每条边，起点/终点各自从自己的 π 抽一个“角色”，再看两角色是否容易连边。能发现“**跨群体的人**”。
 - 缺失 0 的稳健化**：把部分 0 当“未知/没机会”，不强行解释，模型更稳。
 - RTM**：图上的节点是**文档**（有文本）；让“是否有边”由**两文档的主题组合**决定，端到端同时学“文本+链接”，补边更准。
 - 推断与扩展**：与 LDA 类似，常用**变分/Gibbs**；块数未知可用**DP**（无限关系模型）；加**节点特征可做判别式**（相当于“关系版专家混合”）。
-

要点小结 (记忆版)

- 图=离散数据**：用二值稀疏邻接矩阵表示。目标：**发现结构 & 预测连边**。

- **SBM**: $q_i \sim \text{Cat}(\pi)$, 边 $R_{ij} \sim \text{Ber}(\eta_{q_i, q_j})$ 。抓“对外连接模式”的社群；可用变分/折叠 Gibbs； K 不知时 \rightarrow DP (无限关系模型)。
- **带特征的 SBM**: $\Pr(R_{ij} = 1)$ 用 $w_{a,b}^\top f(x_i, x_j)$ (逻辑回归)，是“关系版专家混合”。
- **MMSBM**: 节点有**混合成员** $\pi_i \sim \text{Dir}$; 每条边抽 $q_{i \rightarrow j}, q_{i \leftarrow j}$, 再用 $\eta_{a,b}$ 产边。能识别**摇摆者/多角色**；缺失 0 用 $\rho \delta_0 + (1 - \rho) \text{Ber}$ 稳健化。
- **RTM**: 文本+链接联合建模； $\Pr(R_{ij} = 1) = \sigma(w^\top (q_i \otimes q_j) + w_0)$ 。让**链接信号直连主题选择**，比“LDA 后接逻辑回归”更强 (类比 PLS vs PCA+LR)。

如果你愿意，我可以用一个小玩具图 (十来个节点) 现场演示 SBM / MMSBM 如何把节点“按连接偏好”分组、以及 RTM 如何用主题去预测谁会引用谁。

这一节主要在讲什么？我不太懂

已思考 8s

没关系！这节 (27.5) 讲的是：把“图/网络数据”也当成离散数据来做潜在变量建模，用来做两件事——**找结构** (社群/角色/层级) 和**补边** (预测谁会连谁)。

一句话版

把社交关系、论文引用这类网络用**邻接矩阵** (0/1, 很稀疏) 表示，然后用几类潜在变量模型去：①按“连接偏好”给节点分组；②根据学到的结构预测缺失/未来连边。

三个主角 (直观理解)

1. SBM (随机分块模型)

- 想法：每个节点属于一个块（群体）；“是否连边”只看块对 (a,b) 的连边率 $\eta_{a,b}$ 。
- 重点：同一块的节点不一定互连，但它们对其他块的连接模式相似。
- 训练：用变分 EM / 折叠 Gibbs；块数未知可上 Dirichlet 过程 (“无限关系模型”)。
- 可加**节点特征**：把 $\Pr(R_{ij} = 1)$ 做成逻辑回归 $\sigma(w_{a,b}^\top f(x_i, x_j))$ 。

2. MMSBM (混合成员 SBM)

- 想法：一个人可能**多重身份** (π_i 是各群体占比)。对每条边，起点/终点各自从自己的 π 抽一个角色，看这对角色是否容易连边。
- 能发现“**摇摆者**”/跨圈子的人。
- **缺失 0 的稳健化**：把一部分 0 当成“未知/没机会接触”，不强求模型解释所有 0 ($\rho \delta_0 + (1 - \rho) \text{Bernoulli}$)。

3. RTM (关系主题模型)

- 场景：节点还有**文本属性**（如论文的正文），边是**引用**。
- 做法：一边用 LDA 学每篇的主题分布（经验主题 q_i ），一边用**逻辑回归**根据 $q_i \otimes q_j$ 预测是否有边。
- 关键：让边依赖**实际选到的主题** q （而不是平均的 π ），这样**链接信号**不会被大量词信息稀释，补边更准。

该用谁？

- **只有结构**（没属性）：先试 SBM；若人有多重角色 → MMSBM。
- **有文本+链接一起学**：RTM。
- **块数不确定**：用非参（Dirichlet 过程）版本。
- **有节点特征想进模型**：用带特征的判别式 SBM（关系版“专家混合”）。

要点小结

- 图数据=稀疏邻接矩阵；目标：**找社群/角色 + 预测连边**。
- SBM：单一群体标签，按“块对连边率”刻画结构；可变分/Gibbs、可非参、可用特征。
- MMSBM：多重身份 π ，边级别抽“角色对”，贴近真实网络；缺失 0 可稳健处理。
- RTM：文本与链接**联合建模**，用主题组合直接预测边，优于“先 LDA 后回归”的两步法。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

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:

$$R \subseteq T_1 \times T_2 \times \cdots \times T_k \quad (27.81)$$

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

$\text{seen} = \{(Bob, StarWars), (Bob, TombRaider), (Alice, Jaws)\}$
or implicitly, using an indicator function for the set:
 $\text{seen}(Bob, StarWars)=1$, $\text{seen}(Bob, TombRaider)=1$,
 $\text{seen}(Alice, Jaws)=1$

A relation between two entities of types T_1 and T_2 can be represented as a binary function $R : T_1 \times T_2 \rightarrow \{0, 1\}$, 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, $R : T_1 \times T_1 \times T_2 \rightarrow \{0, 1\}$, 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_t $i \in \{1, \dots, K_t\}$ 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 $R : T_1 \times T_1 \times T_2 \rightarrow \{0, 1\}$, we have

$$p(R(i, j, k)=1 | q_1 i = a, q_1 j = b, q_2 k = c, \eta) = \eta_{a,b,c} \quad (27.82)$$

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 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 $R : T_1 \times T_1 \times T_2 \rightarrow \{0, 1\}$, 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 homogeneous 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 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

$$p(R1, R2, q1, q2, q3 | \theta) = p(R1|q1, q2, \theta)p(R2|q1, q3, \theta) \quad (27.83)$$

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 $R : T1 \times T2 \rightarrow R$, 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:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (X(mi, ui) - \hat{X}(mi, ui))^2} \quad (27.84)$$

where $X(mi, ui)$ is the true rating of user ui on movie mi , and $\hat{X}(mi, ui)$ 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 by 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 $R_{ij} = 6 - R_{ij}$ (Aggarwal and Merugu 2007).

We could use the IRM for the CF task, by associating a discrete latent variable for each user q_{ui} and for each movie or video q_{vj} , and then defining

$$p(R_{ij} = r | q_{ui} = a, q_{vj} = b, \theta) = N(r | \mu_a, \sigma^2) \quad (27.85)$$

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 $\pi_{ui} \in SK_u$ and $\pi_{vj} \in SK_v$. 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 $u_i \in RK$ and each movie $v_j \in RK$.⁷ We then use a likelihood of the form

$$p(R_{ij} = r | u_i, v_j) = N(r | u_i T v_j, \sigma^2) \quad (27.86)$$

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. PMF is closely related to the SVD. In particular, if there is no missing data, then computing the MLE for the u_i 's and the v_j 's is equivalent to finding a rank K approximation to R . However, as soon as we have missing data, the problem becomes non-convex, as shown in (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:

$$J(U,V) = -\log p(R|U,V,O) = -\log \left(\prod_{i=1}^N \prod_{j=1}^M \prod_{O_{ij}=1} N(R_{ij}|u_i^T v_j, \sigma^2) \right) \quad (27.87)$$

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 u_i is given by

$$\frac{\partial J}{\partial u_i} = \sum_j \frac{\partial \log p(R_{ij}|u_i^T v_j, \sigma^2)}{\partial u_i} = -\sum_j O_{ij} (R_{ij} - u_i^T v_j)^2 = -\sum_j O_{ij} e_{ij}^2 \quad (27.88)$$

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

$$u_i = u_i + \eta e_{ij} v_j \quad (27.89)$$

where η is the learning rate. The update for 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:

$$p(U,V) = \prod_i N(u_i|\mu_u, \Sigma_u) \prod_j N(v_j|\mu_v, \Sigma_v) \quad (27.90)$$

If we use $\mu_u = \mu_v = 0$, $\Sigma_u = \sigma^2 I_{IK}$, and $\Sigma_v = \sigma^2 V_{IK}$, the new objective becomes

$$J(U,V) = -\log p(R|U,V|O, \theta) \quad (27.91)$$

$$= \sum_i \sum_j O_{ij} (R_{ij} - u_i^T v_j)^2 + \lambda U \sum_i \|u_i\|^2 + \lambda V \sum_j \|v_j\|^2 + \text{const} \quad (27.92)$$

where we have defined $\lambda U = \sigma^2 / \sigma^2 U$ and $\lambda V = \sigma^2 / \sigma^2 V$.

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 $(\mu_u, \Sigma_u, \mu_v, \Sigma_v)$ at the same time as the model parameters (U, V, σ^2) is a way to create an adaptive prior. This avoids the need to search for the optimal values of λU and λV , 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:

$$p(R_{ij} = r | u_i, v_j, \theta) = N(r | u^T i v_j + \mu + f_i + g_j, \sigma^2) \quad (27.93)$$

where μ is the overall mean, f_i is the user bias, g_j is the movie bias, and $u^T i v_j$ 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 μ terms using stochastic gradient descent, just as we estimated U , V and θ .

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) 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 x_v , the features of the video. (In the case where we just have the id of the video, we can treat x_v as a $|V|$ -dimensional bit vector with just one bit turned on.) We may also know features about the user, which we can denote by 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:

$$p(R(u, v)|x_u, x_v, \theta) = \text{Ber}(R(u, v)|(Ux_u)^T(Vx_v)) \quad (27.94)$$

where U is a $|U| \times K$ matrix, and V is a $|V| \times K$ matrix (we can incorporate an offset term by appending a 1 to x_u and x_v in the usual way). A method for computing the approximate posterior $p(U, V|D)$ 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.

已思考 13s

下面把整段内容逐句完整翻译，并在每个小节后给出通俗、详尽的解释（尽量不依赖数学）。最后附上要点小结。个别排版/符号小问题（如 $R_{ij} = "6 - R_{ij}$ 很可能是 $R_{ij} = 6 - R_{ij}$ ）不影响理解，我会在解释里说明。

27.6 关系数据的潜在变量模型 (LVMs for relational data)

原文翻译

图可以用来表示某一类变量之间的关系数据，例如人和人之间的友谊关系。但我们经常会有**多种对象类型**，以及**多种关系类型**。举例来说，图 27.24 说明了两种关系：一种在人和人之间，另一种在人和电影之间。

一般地，我们把一个 k 元关系 R 定义为适当类型的 k 元组的一个子集：

$$R \subseteq T_1 \times T_2 \times \dots \times T_k \quad (27.81)$$

其中 T_i 是集合或类型。二元/成对/二者 (dyadic) 关系是定义在对象对上的关系。例如，“看过 (seen)”这个人-电影关系可以表示为人看过的电影的集合。我们可以显式地把它写成一个集合，例如

$$\text{seen} = \{(Bob, \text{StarWars}), (Bob, \text{TombRaider}), (Alice, \text{Jaws})\}$$

也可以隐式地用该集合的指示函数来表示：

$$\text{seen}(Bob, \text{StarWars}) = 1, \text{seen}(Bob, \text{TombRaider}) = 1, \text{seen}(Alice, \text{Jaws}) = 1$$

类型为 T_1 与 T_2 的两个实体之间的关系，可以表示为二值函数 $R : T_1 \times T_2 \rightarrow \{0, 1\}$ ，因此也可表示为一个二值矩阵。这也可以表示为二分图，图中有两种类型的节点。若 $T_1 = T_2$ ，就退化为常规的有向图（见 27.5 节）。不过，也有一些情形不易用图来建模，但仍可用关系来建模。例如，我们可能有一个三元关系 $R : T_1 \times T_1 \times T_2 \rightarrow \{0, 1\}$ ，比如当且仅当化学物质 k 存在时，蛋白 i 与蛋白 j 相互作用，就令 $R(i, j, k) = 1$ 。这可以用一个三维的二值矩阵来建模。我们会在 27.6.1 节给出这样的例子。

对关系数据做概率建模称为统计关系学习 (Getoor and Taskar, 2007)。一种做法是用图模型直接建模变量之间的关系，这叫概率关系建模。另一种做法是使用潜在变量模型，如下所述。

27.6.1 无限关系模型 (Infinite relational model, IRM)

原文翻译

把随机分块模型扩展到关系数据是直接的：我们只需给每一类型 t 的每个实体 i 关联一个潜在变量 $q_i^t \in \{1, \dots, K_t\}$ 。然后通过查表（看对应类型的实体之间关系成立的概率）来定义具体实体之间关系成立的概率。比如，若 $R : T_1 \times T_1 \times T_2 \rightarrow \{0, 1\}$ ，我们有

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

如果对每种类型的簇数 K_t 不做上限（用狄利克雷过程），该模型称为无限关系模型 (IRM) (Kemp et al., 2006)。本质上相同的模型，以无限隐式关系模型 (IHRM) 之名同时被 (Xu et al., 2006) 提出。我们可以用变分贝叶斯 (Xu et al., 2006, 2007) 或折叠 Gibbs 采样 (Kemp et al., 2006) 来拟合。我们不展开算法细节（与 LDA 类似），而是概述一些有趣的应用。

27.6.1.1 学习本体 (Learning ontologies)

原文翻译

本体 (ontology) 指知识的组织结构。在 AI 中，本体经常是手工构建的（如 Russell and Norvig, 2010），但从数据中学习本体也很有趣。（Kemp et al., 2006）展示了如何用 IRM 来完成这件事。

数据来自**统一医学语言系统** (UMLS; McCray, 2003)，它定义了一个**语义网络**，含有 135 个**概念**（如“疾病或综合征”“诊断过程”“动物”），以及 49 个**二元谓词**（如“影响 (affects)”“预防 (prevents)”）。我们可以把这表示为一个**三元关系** $R : T_1 \times T_1 \times T_2 \rightarrow \{0, 1\}$ ，其中 T_1 是概念集合， T_2 是二元谓词集合。结果就是一个**三维立方体**。然后我们应用 IRM，把这个立方体切分为**响应大致均匀**的分区。系统找到了 14 个**概念簇**和 21 个**谓词簇**。其中一些见图 27.25。系统例如学到了“**生物功能会影响生物体**”（因为当 a 表示生物功能簇、 b 表示生物体簇、 c 表示 affects 谓词簇时， $\eta_{a,b,c} \approx 1$ ）。

27.6.1.2 基于关系与特征的聚类

原文翻译

我们还可以用 IRM 基于**关系**和**特征**来聚类对象。例如，(Kemp et al., 2006) 考察了一个 1965 年的**政治数据集**，包含 14 个国家、54 个表示国家间交互类型的**二元谓词**（如“送游客到... (sends tourists to)”“经济援助 (economic aid)”），以及 90 个**特征**（如“共产主义”“君主制”）。为了得到二值数据，实值特征以其均值为阈值进行二值化，**类别型变量**做了**哑编码**。该数据有 3 种类型： T_1 表示国家， T_2 表示交互， T_3 表示特征。我们有两个关系： $R_1 : T_1 \times T_1 \times T_2 \rightarrow \{0, 1\}$ ，和 $R_2 : T_1 \times T_3 \rightarrow \{0, 1\}$ 。（因此该问题结合了**双聚类**和**本体发现**两方面。）给定多重关系时，IRM 把它们视为**条件独立**。在此情形下，

$$p(R_1, R_2, q_1, q_2, q_3 | \theta) = p(R_1 | q_1, q_2, \theta) p(R_2 | q_1, q_3, \theta) \quad (27.83)$$

结果见图 27.26。IRM 把 90 个特征分成了 5 个簇，其中**第 1 个簇**包含 “noncommunist (非共产主义)”，捕捉到了这个**冷战时代**数据集的一个关键方面。它还把 14 个国家聚成 5 簇，反映了**自然的地缘政治分组**（如美国与英国、或共产阵营），并把 54 个谓词聚成 18 簇，反映了**相似的关系类型**（如“负面行为”“指责”）。

通俗解释 (整段 27.6.1)

- **IRM 是“多类型对象 + 多关系”的 SBM 升级版**：给每个类型的每个实体一个潜在簇标签（或非参地让簇数可变），再给“簇与簇之间的关系”一个概率表 η ，就能在一个或多个关系上**同时解释**数据。
- **三元关系** (概念×概念×谓词) → 把三维 0/1 立方体按“块”切开；“块里几乎全 1 或全 0”。这就等于**自动学习本体** (概念簇 & 谓词簇)。
- **多关系 + 特征**：国家×国家×交互 & 国家×特征 两个关系一起学，能同时发现**国家分组、交互类型分组、特征分组**；而且会抓住**核心分裂轴** (如冷战阵营)。
- **非参数贝叶斯 (DP) 让簇数自动决定；拟合用变分或折叠 Gibbs**，与 LDA 很像。

27.6.2 用于协同过滤的概率矩阵分解 (PMF for collaborative filtering)

原文翻译

如 1.3.4.2 节所述，**协同过滤 (CF)** 要预测一个矩阵 $R : T_1 \times T_2 \rightarrow \mathbb{R}$ 的条目，例如 $R(i, j)$ 是用户 i 给电影 j 的评分。可见 CF 是一种**关系学习**问题（而且商业上尤为重要）。

该领域大量工作使用了 Netflix 竞赛提供的数据。具体而言，提供了一个**电影×用户评分矩阵**，大小为 $17,770 \times 480,189$ 。完整矩阵有 $\sim 8.6 \times 10^9$ 个条目，但只观测到 100,480,507 个（约 1%），因此矩阵**极度稀疏**。另外数据非常**不平衡**：很多用户只评了不到 5 部电影，少数用户评了超过 10,000 部。验证集有 1,408,395 个（电影, 用户）对。最后还有一个单独的测试集，包含 2,817,131 个（电影, 用户）对，其评分已知但对参赛者**保密**。**性能度量是均方根误差 (RMSE)**：

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

其中 $X(m_i, u_i)$ 是真实评分， $\hat{X}(m_i, u_i)$ 是预测。**基线系统** Cinematch 的训练集 RMSE 为 0.9514，测试集为 0.9525。要拿到大奖，队伍需要把测试 RMSE **降低** 10%，即达到 0.8563 或更低。下面我们讨论获胜队伍使用的一些基本方法。

由于评分来自集合 $\{0, 1, 2, 3, 4, 5\}$ ，很容易想到用**类别观测模型**。但这不能体现评分的**有序性**。虽然我们可以用**有序模型**，但实践中人们往往为简单起见使用**高斯观测模型**。一种让模型更贴近数据的方法是：先把模型预测的**均值**经过一个 sigmoid，再把 $[0, 1]$ 区间映射到 $[0, 5]$ (Salakhutdinov and Mnih, 2008)。或者，我们也可以把数据变得更像高斯：用变换 $R_{ij} = 6 - R_{ij}$ (原文处疑似排版为 $R_{ij} = "6 - R_{ij}$; Aggarwal and Merugu, 2007)。

我们可以用 IRM 做 CF：给每个用户一个离散潜在变量 q_i^u ，给每个电影一个离散潜在变量 q_j^v ，并定义

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

这其实是一种**共同聚类 (co-clustering)**。我们还可以扩展模型去生成**侧信息**，如每个用户/电影的属性。示意见图 27.27。

另一种可能是把离散潜在变量换成**连续潜在变量**： $\pi_i^u \in S_{K_u}$ 、 $\pi_j^v \in S_{K_v}$ 。不过，已经发现（如 Shan and Banerjee, 2010），若对每个用户/电影使用**不受约束的实值潜在因子** $u_i \in \mathbb{R}^K$ 、 $v_j \in \mathbb{R}^K$ ，效果更好。此时似然为

$$p(R_{ij} = r \mid u_i, v_j) = N(r \mid u_i^\top v_j, \sigma^2) \quad (27.86)$$

这称为**概率矩阵分解 (PMF)** (Salakhutdinov and Mnih, 2008)。DGM 见图 27.28(a)。

直觉是：把用户与电影嵌入到同一个低维连续空间（见图 27.28(b)）。如果某用户与某电影在该空间里“靠得近”，他就可能打高分。Netflix 竞赛中所有最佳方案都以此为核心。

PMF 与 SVD 紧密相关。特别地，若**没有缺失数据**，则对 $\{u_i\}$ 、 $\{v_j\}$ 做 MLE 等价于对 R 做秩 K 近似。然而，一旦**有缺失数据**，问题就变成**非凸** (Srebro and Jaakkola, 2003)，标准 SVD 不能直接用。（回忆 Netflix 挑战只有约 1% 的矩阵被观测。）

拟合 PMF 的最直接方式是**最小化负对数似然 (NLL)**：

$$J(U, V) = -\log p(R \mid U, V, O) = -\log \left(\prod_{i=1}^N \prod_{j=1}^M N(R_{ij} \mid u_i^\top v_j, \sigma^2)^{\mathbf{I}(O_{ij}=1)} \right) \quad (27.87)$$

其中 $O_{ij} = 1$ 表示用户 i 看过电影 j 。由于该问题**非凸**，我们寻找**局部最优**的 MLE。Netflix 数据很大（约一亿个观测条目），常用**随机梯度下降 (SGD)** (§8.5.2)。对 u_i 的梯度为

$$\frac{\partial J}{\partial u_i} = \frac{\partial}{\partial u_i} \frac{1}{2} \sum_j \mathbf{I}(O_{ij} = 1) (R_{ij} - u_i^\top v_j)^2 = - \sum_{j: O_{ij}=1} e_{ij} v_j \quad (27.88)$$

其中 $e_{ij} = R_{ij} - u_i^\top v_j$ 是误差项。随机抽取用户 i 看过的一部电影 j ，更新为

$$u_i \leftarrow u_i + \eta e_{ij} v_j \quad (27.89)$$

其中 η 为学习率。 v_j 的更新类似。

当然，只最大化似然会**过拟合**（见图 27.29(a)）。我们可以用**高斯先验**来正则化：

$$p(U, V) = \prod_i N(u_i \mid \mu_u, \Sigma_u) \prod_j N(v_j \mid \mu_v, \Sigma_v) \quad (27.90)$$

若取 $\mu_u = \mu_v = 0$ 、 $\Sigma_u = \sigma_u^2 I_K$ 、 $\Sigma_v = \sigma_v^2 I_K$ ，新的目标为

$$\begin{aligned} J(U, V) &= -\log p(R, U, V \mid O, \theta) \\ &= \sum_{i,j} \mathbf{I}(O_{ij} = 1) (R_{ij} - u_i^\top v_j)^2 + \lambda_U \sum_i \|u_i\|_2^2 + \lambda_V \sum_j \|v_j\|_2^2 + \text{const} \end{aligned} \quad (27.92)$$

其中 $\lambda_U = \sigma^2 / \sigma_u^2$ 、 $\lambda_V = \sigma^2 / \sigma_v^2$ 。通过调节正则化，可缓解过拟合（见图 27.29(a)）。我们可用 SGD 求 MAP，也可用**变分贝叶斯近似后验** (Iljin and Raiko, 2010)。

若对先验使用**对角协方差**，就能对每个潜在维度**施加不同惩罚**。若使用**非零均值先验**，就能吸收**偏置项**。同时优化先验参数 $(\mu_u, \Sigma_u, \mu_v, \Sigma_v)$ 与模型参数 (U, V, σ^2) 形成**自适应先验**，避免手工搜 λ_U, λ_V 的最优值，并带来更好效果（见图 27.29(a)）。

事实证明，数据中很大一部分方差来自**电影特定或用户特定的偏置效应**。例如，有些电影对各种用户都很受欢迎；有些用户对各种电影都打低分。我们可以在模型里加入用户/电影偏置：

$$p(R_{ij} = r \mid u_i, v_j, \theta) = N(r \mid u_i^\top v_j + \mu + f_i + g_j, \sigma^2) \quad (27.93)$$

其中 μ 是总体均值， f_i 是用户偏置， g_j 是电影偏置， $u_i^\top v_j$ 是交互项。等价于仅对**残差矩阵**做 PMF，能显著提升效果（见图 27.29(b)）。 μ, f_i, g_j 可像 U, V, θ 一样用 SGD 估计。

我们还可让偏置项**随时间演化**，以反映用户偏好变化（Koren, 2009b）。这很重要，因为 Netflix 测试数据比训练数据**更近时**。图 27.29(b) 显示考虑时间动态帮助很大。

我们经常还拥有各种**侧信息**。在 Netflix 竞赛中，参赛者知道用户在测试集中**评过哪些电影**，尽管不知道评分数值。也就是说，他们知道测试集上**稠密**的 O 矩阵。若用户选择去评分，通常因为他看过电影，且**预期会喜欢**；评不评本身就传递了信息。反过来，未评分暗示他可能知道自己**不喜欢**。因此数据**并非随机缺失 (MNAR)**（见 Marlin and Zemel, 2009）。利用这一点能提升性能（见图 27.29(b)）。在真实问题中，测试集信息并不可得，但我们常常知道用户**看过/拒看了哪些电影 (隐式反馈)**，这也是有用的侧信息。

另一类侧信息是电影的**内容**，如类型、演员表、剧情简介，可用 x_v 表示电影特征。（如果只有电影 ID，我们可把 x_v 当作 $|V|$ 维的**one-hot** 向量。）我们也可能知道用户特征 x_u 。有时我们只知道用户**点没点该视频**（没有数值评分）。这时我们可以把模型改为

$$p(R(u, v) \mid x_u, x_v, \theta) = \text{Ber}(R(u, v) \mid (Ux_u)^\top (Vx_v)) \quad (27.94)$$

其中 U 是 $|U| \times K$ 矩阵， V 是 $|V| \times K$ 矩阵（可像常规做法那样在 x_u, x_v 末尾拼 1 来吸收偏置）。一种用 ADF 与 EP 做**在线近似后验** $p(U, V \mid D)$ 的方法见（Stern et al., 2009）。该方法由微软实现并部署，用于预测 Bing 全量广告的**点击率 (CTR)**。

不幸的是，仅用**正例二值**数据来拟合该模型，**会过度预测链接**，因为没有负例。若能拿到“用户看见过哪些视频、其中至多选择了一个”的日志（**曝光日志**，impression log），效果会更好。在这种情况下，可用**多项式模型**代替二元模型；（Yang et al., 2011）显示这种做法明显优于二元模型。直觉：假设给用户呈现三选——施瓦辛格主演的动作片、范迪塞尔主演的动作片、休·格兰特主演的喜剧；若用户选了施瓦辛格，我们不只学到他**更偏好动作片胜于喜剧**，还学到他**更偏好施瓦辛格胜于迪塞尔**。这比只知道“喜欢施瓦辛格和动作片”**信息量大得多**。

通俗解释 (整段 27.6.2)

- 核心：**把用户和物品各自学成一个低维向量 u_i, v_j ，用它们的内积 $u_i^\top v_j$ 来预测评分（或点击/选择概率）。

- **训练：**只有 1% 的观测 → 直接 SVD 不行，改用 SGD 最小化只在观测点上的平方误差；加上 L_2 正则（即高斯先验）防过拟合；再加上**用户/物品偏置与时间漂移**会更准。
 - **侧信息：**
 - **观测指示** O 透露“看/不看/评/不评”的**选择偏差** (MNAR)，利用它能提升效果；隐式反馈同理可用。
 - **内容特征** x_u, x_v 可用到“点不点/选哪个”上（逻辑回归式），在线近似推断可扩展到大规模工业系统（如 Bing CTR）。
 - **曝光级 (多项选择) 建模**胜过“只看正例”的二元建模，因其携带相对偏好信息。
-

要点小结 (记忆版)

- **关系数据：**不仅有人-人这种单一图，还可能有多类型对象、多关系（人-电影、人-人、概念-概念-谓词...）。形式化为 $R \subseteq T_1 \times \dots \times T_k$ 。
- **IRM (无限关系模型) :** 多类型版本的 SBM。给每种类型的实体分簇（簇数可用 DP 自适应），关系由“**簇-簇 (-簇) 概率表**”决定。可
 - **学本体：**把 3D 关系立方体切成“高/低概率块”，得到**概念簇+谓词簇**；
 - **多关系+特征联合聚类：**同时得到**国家/交互/特征**的簇，抓住数据的主轴（如冷战阵营）。
- **CF→PMF:** 用户/电影嵌入到同一低维空间，评分≈**内积**；SGD + L_2 正则；再加**偏置项**（用户/电影/时间）显著降误差。
- **侧信息与缺失机制：**
 - 观测指示 O 暗含**选择偏差 (MNAR)**，应建模（隐式反馈亦然）；
 - 加入内容特征 x_u, x_v 可做点击/选择预测（伯努利/多项）；
 - **曝光日志 (impression log) 上的多项选择建模**能学习到**相对偏好**，优于仅正例的二元建模。

小提示：文本中的 $R_{ij} = "6 - R_{ij}$ 可理解为把评分做线性变换 $R' = 6 - R$ ，让数据更接近高斯假设；实际系统常用**归一化/对数/Box-Cox** 等变换与偏置/时间项配合以提升拟合。

这一节主要在讲什么？我不太懂

已思考 9s

没问题！这节（27.6）的主线是：**怎么用潜在变量模型来处理“关系型数据”**——不只是
一张单一类型的图（人 \rightarrow 人），还包含**多种对象类型**（人、电影、概念...）和**多种关系**（人看
过电影、概念影响概念、国家对国家的互动等）。

一句话版

把“谁和谁有什么关系”抽象成一个（可能是二元、三元...）的**关系表/张量**，再用潜在变量
(比如“每个实体属于哪个隐含簇”“每个实体的低维向量”)去解释、压缩和预测这些关系。

这节都讲了什么（通俗概览）

- **关系数据的形式化**

不同类型的对象 T_1, \dots, T_k 之间的 k 元关系 $R \subseteq T_1 \times \dots \times T_k$ 。

二元关系可以看成**二值矩阵或二分图**；三元关系可以看成**三维二值张量**（比如“蛋白i
与蛋白j 在 化学物k 下是否相互作用”）。

- **IRM：无限关系模型（27.6.1）**

多类型版的 SBM：

- 给**每种类型**的每个实体一个**隐含簇标签**（簇数用狄利克雷过程自动决定，不用手
调）。

- 关系是否成立由“**簇 \times 簇 \times 簇**”的概率表”决定。

- 能同时处理**多种关系**（如 国家 \times 国家 \times 互动、国家 \times 特征），并把对象、关系类
型、特征**各自分簇**。

- 应用示例：

- **学本体**：把“概念 \times 概念 \times 谓词”的三维立方体切成高/低概率块，学出**概念簇**、
谓词簇及它们的对应关系。

- **关系+特征联合聚类**：冷战数据里同时得出国家分组、互动分组、特征分
组，还能自动抓住“非共产主义”等关键分裂轴。

- **PMF：概率矩阵分解（27.6.2）**（协同过滤的主力）

- 场景：**用户 \times 物品评分矩阵**极稀疏（如 Netflix）。

- 思路：给每个用户/物品一个**低维实向量** u_i, v_j ，用**内积** $u_i^\top v_j$ 预测评分（或点击/
选择概率）。

- 训练：只在观测位置上最小化平方误差，配合 SGD 与 L_2 正则（等价高斯先验）
避免过拟合。

- **偏置与时间**：加入**全局/用户/物品偏置**，再让偏置**随时间漂移**（口味变化） \rightarrow 大
幅降误差。

- **侧信息与缺失机制**：

- 评分是否出现本身就携带信息（**非随机缺失 MNAR**、**隐式反馈**）；建模可提升效果。
- 可融合**内容特征**（类型、演员、简介、用户属性）做点击/选择预测（伯努利或**多项选择**建模）。
- 有**曝光日志**时，用“候选集中选1”的**多项模型**比二元正例更强，因为能学到**相对偏好**（更信息量）。

何时用哪种？

- 想发现**关系结构 / 自动分层、分簇**（还能处理多种对象与多种关系）→ 用 **IRM**。
- 想做**评分/点击等精确预测**（特别是超稀疏的大矩阵）→ 用 **PMF**（矩阵分解）+ 正则 + 偏置 + 时间/侧信息。
- **只有二元链接且有实体特征** → 可用 **IRM** 的判别式变体或“关系版专家混合”（逻辑回归到链接概率）。

要点小结

- 关系数据 ≠ 只是一张图；它可以是多类型、多关系，形式化成 $R \subseteq T_1 \times \dots \times T_k$ 。
- **IRM**：给各类型实体分隐簇，关系由簇间概率决定；支持**多关系与非参簇数**；可用于**学本体、联合聚类**。
- **PMF**：把用户/物品嵌入同一低维空间，评分≈**向量内积**；用 **SGD + 正则** 训练，加入**偏置与时间动态、侧信息、多项选择**能显著提升表现。

如果你手头有具体的数据（比如一张用户-商品的交互表、或多关系的实体表），我可以按这个思路帮你选模型、搭个最小可用原型。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

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 h and visible nodes 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

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

where R is the number of visible (response) variables, K is the number of hidden variables, and v plays the role of 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) 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 1 or 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" (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 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 $q \in \{1, \dots, K\}$. 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 $\pi \in SK$, as in mPCA/ LDA, or $z \in RK$, 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:

$$p(h|v, \theta) = \prod_k p(h_k|v, \theta) \quad (27.96)$$

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

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

$$\begin{aligned} E(v, h; \theta) &= -\sum_r \sum_k v_r h_k W_{rk} - \sum_r v_r b_r - \sum_k h_k c_k \\ &= -(v^T W h + v^T b + h^T c) \end{aligned} \quad (27.98)$$

$$Z(\theta) = \sum_v \sum_h \exp(-E(v, h; \theta)) \quad (27.100)$$

where E is the energy function, W is a $R \times K$ weight matrix, b are the visible bias terms, c are the hidden bias terms, and $\theta = (W, b, c)$ 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 $w_{0,:} = c$ and $w,:0 = b$. Note that naively computing $Z(\theta)$ takes $O(2RK)$ time but we can reduce this to $O(\min\{R^2K, K^2R\})$ time (Exercise 27.1).

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

$$p(h|v, \theta) = \prod_k p(h_k|v, \theta) = \prod_k \text{Ber}(h_k | \text{sigm}(w^T v + b_k)) \quad (27.101)$$

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

$$p(v|h, \theta) = \prod_r p(v_r|h, \theta) = \prod_r \text{Ber}(v_r | \text{sigm}(w^T r + b)) \quad (27.102)$$

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

$$E[h|v\theta] = \text{sigm}(WTv) \quad (27.103)$$

$$E[v|h, \theta] = \text{sigm}(Wh) \quad (27.104)$$

The weights in W are called the generative weights, since they are used to generate the observations, and the weights in WT 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 v "looks like" the weight vector $w_{:,k}$ (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 vir. We define a new energy function as follows (Salakhutdinov et al. 2007; Salakhutdinov and Hinton 2010):

$$E(v, h; \theta) = - \sum_r R r=1 \sum_k K k=1 \sum_c C c=1 v_c r h_k W_{c,r} k - \sum_r R r=1 \sum_c C c=1 v_c r b_c r - \sum_k K k=1 h_k c_k (27.105)$$

The full conditionals are given by

$$p(v_r|h, \theta) = \text{Cat}(S(\{b_c r + h_k W_{c,r} k\}C c=1)) \quad (27.106)$$

$$p(h_k = 1|c, \theta) = \text{sigm}(c_k + r c v_c r W_{c,r} k) \quad (27.107)$$

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:

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

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

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

where $\eta = \Lambda \mu$. We see that we have set $\Lambda = I$, and $\eta = k h_{k,:}$. Thus the mean is given by $\mu = \Lambda^{-1} \eta = k h_{k,:}$.

The full conditionals, which are needed for inference and learning, are given by

$$p(v_r|h, \theta) = N(v_r|b_r + k w_{rk}, 1) \quad (27.110)$$

$$p(h_k = 1|v, \theta) = \text{sigm}(\eta_k + r v_r W_{r,k}) \quad (27.111)$$

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 L_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 $p(h, v|\theta)$

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

$$\partial \ell / \partial w_{rk} = 1/N \sum_{i=1}^N E[v_r h_k | v_i, \theta] - E[v_r h_k | \theta] \quad (27.112)$$

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

$$\nabla w_l = E[p_{emp}(\cdot | \theta) v h^T] - E[p(\cdot | \theta) v h^T] \quad (27.113)$$

where $p_{emp}(v, h | \theta) = p(h | v, \theta) p_{emp}(v)$, and $p_{emp}(v) = 1/N \sum_{i=1}^N \delta(v_i(v))$ 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 v is fixed to a data case, is sometimes called the clamped phase, and the second term, when 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 $p(v|\theta)$

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 $p(v|\theta) = \frac{1}{Z(\theta)} \exp(-F(v; \theta))$, where $F(v; \theta)$ is the free energy:

$$F(v) = -\sum_h E(v, h) = -\sum_h \exp(-F(v; \theta)) \quad (27.114)$$

$$= -\sum_h \exp(-F(v; \theta)) \quad (27.115)$$

$$= -\sum_h \exp(-F(v; \theta)) \quad (27.116)$$

$$= -\sum_h \exp(-F(v; \theta)) \quad (27.117)$$

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

$$I(\theta) = -\sum_i N \log p(v_i|\theta) = -\sum_i N \log F(v_i|\theta) + \log Z(\theta) \quad (27.118)$$

Using the fact that $Z(\theta) = \sum_v \exp(-F(v; \theta))$ we have

$$\nabla I(\theta) = -\sum_i N \nabla \log F(v_i) = -\nabla \log Z(\theta) \quad (27.119)$$

$$= -\sum_i N \nabla \log \sum_v \exp(-F(v; \theta)) = -\sum_i N \nabla \log \sum_v \exp(-F(v; \theta)) \quad (27.120)$$

$$= -\sum_i N \nabla \log \sum_v \exp(-F(v; \theta)) + \nabla \log \sum_v \exp(-F(v; \theta)) \quad (27.121)$$

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

$$\partial \nabla I(\theta) = -\sum_i N \nabla \log F(v_i|\theta) = -\sum_i N \nabla \log \exp(-F(v_i|\theta)) = -\sum_i N \nabla F(v_i|\theta) \quad (27.122)$$

Hence

$$\partial \nabla I(\theta) = -\sum_i N \nabla \log F(v_i|\theta) = -\sum_i N \nabla \log \exp(-F(v_i|\theta)) = -\sum_i N \nabla F(v_i|\theta) \quad (27.123)$$

which matches Equation 27.112.

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 $p(v, h|\theta)$ as follows: initialize the chain at v_1 (e.g. by setting $v_1 = v$ for some data vector), and then sample from $h_1 \sim p(h|v_1)$, then from $v_2 \sim p(v|h_1)$, then from $h_2 \sim p(h|v_2)$, 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 $E[vrhk] \approx E[vr] E[hk]$. However, since $p(v, h)$ 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 $E[vr_{hk}|v] - E[vr_{hk}]$, 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:

$$\nabla w_l \approx E[vhT|v_i] - Eq[vhT] \quad (27.124)$$

where q corresponds to the distribution generated by K up-down Gibbs sweeps, started at v_i , as in Figure 27.31.

This is known as CD-K. In more detail, the procedure (for $K = 1$) is as follows:

$$h_i \sim p(h|v_i, \theta) \quad (27.125)$$

$$v_i \sim p(v|h_i, \theta) \quad (27.126)$$

$$h_i \sim p(h|v_i, \theta) \quad (27.127)$$

We then make the approximation

$$Eq[vhT] \approx v_i(h_i)T \quad (27.128)$$

Such samples are sometimes called fantasy data. We can think of v_i as the model’s best attempt at reconstructing 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 $E[h|v_i]$ instead of a sampled value h_i in the final upwards pass, since this reduces the variance. However, it is not valid to use $E[h|v_i]$ instead of sampling $h_i \sim p(h|v_i)$ 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).

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.

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 for the fact that the observed word-count vector v is larger in magnitude:

$$E(v, h; \theta) = -K \sum_{k=1}^C c_k \ln v_{ch_k} - C \sum_{c=1}^V v_{cb_c} r_c - L \sum_{k=1}^K h_{ck}$$

(27.129)

where $v_c = \prod_{l=1}^L l^{y_{il} = c}$. This is like having a single multinomial node (so we have dropped the r subscript) with C states, where C is the number of words in the vocabulary. This is called the replicated softmax model (Salakhutdinov and Hinton 2010), and is an undirected alternative to mPCA/ LDA.

We can compare the modeling power of RBMs vs LDA by measuring the perplexity on a test set. This can be approximated using annealing importance sampling (Section 24.6.2). The results are shown in Figure 27.32. We see that the LDA is significantly better than a unigram model, but that an RBM is significantly better than LDA. Another advantage of the LDA is that inference is fast and exact: just a single matrix-vector multiply followed by a sigmoid nonlinearity, as in Equation 27.107. In addition to being faster, the RBM is more accurate. This is illustrated in Figure 27.33, which shows precision-recall curves for RBMs and LDA on two different corpora. These curves were generated as follows: a query document from the test set is taken, its similarity to all the training documents is computed, where the similarity is defined as the cosine of the angle between the two topic vectors, and then the top M documents are returned for varying M . A retrieved document is considered relevant if it has the same class label as that of the query's (this is the only place where labels are used).

27.7.3.2 RBMs for collaborative filtering

RBM have been applied to the Netflix collaborative filtering competition (Salakhutdinov et al. 2007). In fact, an RBM with binary hidden nodes and categorical visible nodes can slightly outperform SVD. By combining the two methods, performance can be further improved. (The winning entry in the challenge was an ensemble of many different types of model (Koren 2009a).)

已思考 9s

下面把整段内容逐句完整翻译，并在各小节后配上通俗、详尽的解释（尽量少用数学）。最后附上要点总结。文中个别地方有轻微笔误（我会在解释里指出，不影响理解）。

27.7 受限玻尔兹曼机 (RBMs)

原文翻译

到目前为止，本章我们提出的所有模型都可以用**有向图模型**来表示。但有些模型用**无向图**来表示更合适。比如，玻尔兹曼机 (Boltzmann machine; Ackley et al., 1985) 就是一个**成对马尔可夫随机场 (MRF)**，带有**隐变量** h 与**可见变量** v ，如图 27.30(a) 所示。玻尔兹曼机的主要问题在于：**精确推断**是不可行的，即便使用例如 Gibbs 采样等近似推断也可能很慢。然而，假设我们限制其结构，让结点按层排列，且**同一层内部没有连接**（见图 27.30(b)）。那么模型具有如下形式

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

其中 R 是可见 (响应) 变量的个数， K 是隐变量个数，且此处的 v 在本章前面相当于 y 。这种模型称为**受限玻尔兹曼机 (RBM)** (Hinton, 2002)，或 harmonium (Smolensky, 1986)。

RBM 是**专家乘积 (PoE)** (Hinton, 1999) 的一个特例；之所以这么叫，是因为我们把一组“专家”(此处即每条边上的势函数)相乘后再归一化；而在专家混合 (mixture of experts) 里，我们是对已归一化的分布做**凸组合**。PoE 之所以可能比混合更有效的直观原因是：**每个专家可以施加一个约束** (如果该专家的取值是 0 或 1)，或者施加“**无所谓**”的条件 (取值为 1)。通过把这些专家以不同方式相乘，我们可以得到对满足指定约束的数据有“**尖锐**”预测的分布 (Hinton & Teh, 2001)。举个**文本的分布式建模**例子：某篇文档可能同时有“政府”“黑手党”“花花公子”三个主题。如果我们把每个主题的预测“相乘”，模型可能会对“Berlusconi (贝卢斯科尼)”这个词给出**非常高的概率** (Salakhutdinov & Hinton, 2010)。相比之下，把专家“相加”只会让分布**更宽** (见图 14.17)。

RBM 的隐结点通常是**二值的**，因此 h 指定了哪些约束是**激活的**。把它与我们之前讨论的有向模型作比较是值得的：在**混合模型**里，我们有一个隐变量 $q \in \{1, \dots, K\}$ 。我们可以用 K 个比特来表达它，但要求**恰好一个**比特为 1——这叫**局部式编码 (localist encoding)**，因为**只有一个**隐单元被用来生成响应向量。这类似“大脑里只识别一种对象的**祖母细胞**”的假想。相反，RBM 使用**分布式编码**，**多个**单元共同参与生成每个输出。使用向量隐变量的模型也属于分布式编码，例如 $\pi \in S_K$ (mPCA/LDA) 或 $z \in \mathbb{R}^K$ (ePCA)。

RBM 与有向两层模型的**主要不同**是：在给定可见变量时，**隐变量条件独立**，因此后验可分解：

$$p(h \mid v, \theta) = \prod_k p(h_k \mid v, \theta) \quad (27.96)$$

这使推断比在有向模型中简单得多，因为我们可以像前馈神经网络那样，并行独立地估计每个 h_k 。劣势是：训练无向模型更难，我们稍后会讨论。

27.7.1 RBM 的多种变体

原文翻译

本节我们通过定义不同的成对势函数来描述各种 RBM 形式。见表 27.2 的总结。这些都是指数族 harmonium (Welling et al., 2004) 的特例。

27.7.1.1 二值 RBM

最常见的 RBM 具有二值隐结点与二值可见结点。其联合分布为

$$p(v, h \mid \theta) = \frac{1}{Z(\theta)} \exp(-E(v, h; \theta)) \quad (27.97)$$

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

$$= -(v^\top Wh + v^\top b + h^\top c) \quad (27.99)$$

$$Z(\theta) = \sum_v \sum_h \exp(-E(v, h; \theta)) \quad (27.100)$$

其中 E 是能量函数， W 是 $R \times K$ 的权重矩阵， b 是可见层偏置， c 是隐层偏置， $\theta = (W, b, c)$ 为全部参数。为简化记号，我们可把偏置吸收到权重里：钳住（固定）虚拟单元 $v_0 = 1, h_0 = 1$ ，并令 $w_{0,:} = c, w_{:,0} = b$ 。注意：直接计算 $Z(\theta)$ 的复杂度是 $O(2^R 2^K)$ ，但可以把它降到 $O(\min\{R 2^K, K 2^R\})$ (练习 27.1)。

使用二值 RBM 时，后验可计算为：

$$p(h \mid v, \theta) = \prod_k p(h_k \mid v, \theta) = \prod_k \text{Ber}(h_k \mid \sigma(w_{:,k}^\top v)) \quad (27.101)$$

由对称性可得，给定隐变量生成数据为：

$$p(v \mid h, \theta) = \prod_r p(v_r \mid h, \theta) = \prod_r \text{Ber}(v_r \mid \sigma(w_{r,:}^\top h)) \quad (27.102)$$

用矩阵-向量记法：

$\mathbb{E}[h \mid v, \theta] = \sigma(W^\top v)$ (27.103)
 $\mathbb{E}[v \mid h, \theta] = \sigma(Wh)$ (27.104)

其中 W 里的权重叫**生成权重**（用来生成观测）， W^\top 里的叫**识别权重**（用来“识别”输入）。

由式 (27.101) 可见：隐单元 k 的激活强度**正比于**输入向量 v 与其权重向量 $w_{:,k}$ 的“相似度”（差个尺度因子）。因此每个隐单元像前馈网络那样，**编码**输入的某些“**特征**”。

27.7.1.2 类别型 RBM

我们可以把二值 RBM 扩展到**类别型**可见变量，方法是对每个 v_r 用 1-of- C 编码 (C 是其可能状态数)。新的能量函数为 (Salakhutdinov et al., 2007; Salakhutdinov & Hinton, 2010)：

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

条件分布为

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

$$p(h_k = 1 \mid v, \theta) = \sigma(c_k + \sum_r v_r^c W_{rk}^c) \quad (27.107)$$

27.7.1.3 高斯 RBM

可推广到**实值数据**。具体地，高斯 RBM 的能量是：

$$E(v, h \mid \theta) = - \sum_{r,k} W_{rk} h_k v_r - \frac{1}{2} \sum_r (v_r - b_r)^2 - \sum_k a_k h_k \quad (27.108)$$

参数为 $\theta = (w_{rk}, a_k, b_r)$ 。（假设数据已标准化，方差固定为 $\sigma^2 = 1$ 。）与**信息形式的高斯**比较：

$$N_c(v \mid \eta, \Lambda) \propto \exp(\eta^\top v - \frac{1}{2} v^\top \Lambda v) \quad (27.109)$$

其中 $\eta = \Lambda \mu$ 。可见这里取 $\Lambda = I$ ， $\eta = \sum_k h_k w_{:,k}$ ，因此均值 $\mu = \sum_k h_k w_{:,k}$ 。需要用于推断/学习的条件分布是

$$p(v_r \mid h, \theta) = N(v_r \mid b_r + \sum_k w_{rk} h_k, 1) \quad (27.110)$$

$$p(h_k = 1 \mid v, \theta) = \sigma(a_k + \sum_r w_{rk} v_r) \quad (27.111)$$

可见每个可见单元是一个**高斯**，其**均值**由隐比特向量决定。更强的模型可以让**（协）方差**也依赖隐状态 (Ranzato & Hinton, 2010)。

27.7.1.4 具有高斯隐单元的 RBM

若隐与可见都是高斯，会得到无向版因子分析。不过它与标准有向版是等价的（Marks & Movellan, 2001）。

若隐是高斯、可见是类别型，就得到无向版的分类 PCA (§27.2.2)。 (Salakhutdinov et al., 2007) 把它用于 Netflix 协同过滤，但发现显著不如使用二值隐变量（表达力更强）。

27.7.2 学习 RBM

原文翻译

本节讨论用梯度方法来做 RBM 的极大似然 (ML) 参数估计。由于 RBM 往往参数多、数据大，常用随机梯度下降 (SGD)。此外，标准做法会用 l_2 正则（在本上下文常叫 weight decay）；对目标与梯度的改动很小（见 §8.3.6）。

27.7.2.1 用 $p(h, v | \theta)$ 推导梯度

要计算梯度，我们可改写 §19.5.2 的公式（那部分讲如何拟合一般的潜变量最大熵模型）。在玻尔兹曼机里，每条边对应一个“特征”，于是梯度为

$$\frac{\partial \ell}{\partial w_{rk}} = \frac{1}{N} \sum_{i=1}^N (\text{E}[v_r h_k | v^{(i)}, \theta] - \text{E}[v_r h_k | \theta]) \quad (27.112)$$

用矩阵-向量形式：

$$\nabla_w \ell = \text{E}_{p_{\text{emp}}(\cdot | \theta)}[vh^\top] - \text{E}_{p(\cdot | \theta)}[vh^\top] \quad (27.113)$$

其中 $p_{\text{emp}}(v, h | \theta) = p(h | v, \theta)p_{\text{emp}}(v)$ ，而 $p_{\text{emp}}(v) = \frac{1}{N} \sum_{i=1}^N \delta_{v^{(i)}}(v)$ 是经验分布。（对偏置项，可把 $v_r = 1$ 或 $h_k = 1$ 代入得到类似表达。）

梯度的第一项（固定 v 为数据样本时）常叫夹紧阶段（clamped phase），第二项（ v 自由）叫非夹紧阶段（unclamped phase）。当模型期望等于经验期望，二者相消，梯度为 0，学习停止。该算法最早由 Ackley et al. (1985) 提出。主要难点是高效计算这些期望，下面讨论几种方法。

27.7.2.2 用 $p(v | \theta)$ 推导梯度

给出另一种推导（也适用于其他能量模型）。先把隐变量积分掉，写成

$$p(v | \theta) = \frac{1}{Z(\theta)} \exp(-F(v; \theta)),$$

其中 $F(v; \theta)$ 是自由能：

$$F(v) = -\log \sum_h \exp(\sum_{r,k} v_r h_k W_{rk}) = \sum_k \log(1 + \exp(\sum_r v_r W_{rk}))$$

(以上给出了化简步骤。)

把 (按样本均值的) 对数似然写为

$$\ell(\theta) = \frac{1}{N} \sum_i \log p(v^{(i)} | \theta) = -\frac{1}{N} \sum_i F(v^{(i)} | \theta) - \log Z(\theta) \quad (27.118)$$

利用 $Z(\theta) = \sum_v \exp(-F(v; \theta))$, 有

$$\nabla \ell(\theta) = -\frac{1}{N} \sum_i \nabla F(v^{(i)}) + E_{p(v)}[\nabla F(v)] \quad (27.119-121)$$

代入自由能可得

$$\frac{\partial}{\partial w_{rk}} F(v) = -v_r E[h_k | v, \theta] = -E[v_r h_k | v, \theta] \quad (27.122)$$

因此

$$\frac{\partial \ell}{\partial w_{rk}} = \frac{1}{N} \sum_i E[v_r h_k | v^{(i)}, \theta] - E[v_r h_k | \theta] \quad (27.123)$$

与式 (27.112) 一致。

27.7.2.3 近似这些期望

可以用**分块 Gibbs 采样**近似梯度中的期望, 利用 (27.101) 与 (27.102)。更具体地, 从联合分布 $p(v, h | \theta)$ 采样如下: 把马氏链初始化在 $v^{(1)}$ (例如取某个数据样本 v_i) , 然后采样 $h^{(1)} \sim p(h | v^{(1)})$, 接着 $v^{(2)} \sim p(v | h^{(1)})$, 再 $h^{(2)} \sim p(h | v^{(2)})$, 以此类推 (如图 27.31)。但需要注意, 必须等链到达**平衡分布** (烧入) 后, 样本才可视为来自目标分布, 这可能很慢。

更快的替代是**均值场**: 近似 $E[v_r h_k] \approx E[v_r]E[h_k]$ 。然而, 由于 $p(v, h)$ 通常是**多峰的**, 这种近似通常很差 (会把不同模态平均, 见 §21.2.2)。还有更微妙的原因: 梯度是 $E[v_r h_k | v] - E[v_r h_k]$ 的形式, 前面有负号, 算法会倾向把变分下界**放松** (Salakhutdinov & Hinton, 2009) , 这解释了早期把均值场用于玻尔兹曼机学习 (如 Kappen & Rodriguez, 1998) **效果不佳**。

27.7.2.4 对比散度 (CD)

用 Gibbs 采样算梯度太慢。下面给出更快的方法——**对比散度 (CD)** (Hinton, 2002)。CD 最初是由“两个 KL 散度之差”的目标近似导出的, 而不是直接最大化似然。但从算法角

度看，它类似**随机梯度**：用**简短的**Gibbs采样来近似“非夹紧”期望，并且**把每条链都从真实数据点开始**。即，对单个样本的梯度近似为

$$\nabla_w \ell \approx E[vh^\top | v_i] - E_q[vh^\top] \quad (27.124)$$

其中 q 是从 v_i 开始进行K次上下(up-down) Gibbs扫描得到的分布(见图27.31)，称为CD-K。更具体地(以 $K=1$ 为例)：

$h_i \sim p(h | v_i, \theta) \quad (27.125)$
 $\tilde{v}_i \sim p(v | h_i, \theta) \quad (27.126)$
 $\tilde{h}_i \sim p(h | \tilde{v}_i, \theta) \quad (27.127)$

然后做近似

$$E_q[vh^\top] \approx \tilde{v}_i \tilde{h}_i^\top \quad (27.128)$$

这些样本有时叫**幻想数据(fantasy data)**。可以把 \tilde{v}_i 看作：模型把 v_i 编码再解码后的重构。这和**自编码器**的训练很像(§28.3.2)。

实践中，最后一次向上传播常用 $E[h | \tilde{v}_i]$ 来替代采样的 \tilde{h}_i ，以降低方差。但在之前的向上传播中，**不能用** $E[h | v_i]$ 来替代采样的 h_i ，否则每个隐单元能传递**超过1bit**信息，瓶颈就不“窄”了。

完整过程见算法3。(注意我们沿**正梯度**方向更新，因为在最大化似然。)有多种技巧加速，如**动量**(§8.3.2)、**小批量**、**更新平均**等，细节见Hinton(2010); Swersky et al.(2010)。

27.7.2.5 持续CD(PCD)

§19.5.5中我们介绍了**随机极大似然(SML)**来拟合**最大熵模型**：利用参数变化缓慢这一事实，不必每次迭代都把MCMC跑到收敛(Younes, 1989)。换言之，存在两个**不同时间尺度**的动力学：状态变化快，参数变化慢。该算法在(Tieleman, 2008)中被独立“再发现”，命名为**持续对比散度(PCD)**。伪代码见算法3。

PCD往往比CD更好(如Tieleman, 2008; Marlin et al., 2010; Swersky et al., 2010)，尽管在训练早期CD可能更快。

27.7.3 RBM的应用

原文翻译

RBM的主要应用是作为**深层生成模型的构建块**(§28.2)。但它们也可以**替代**有向两层模型。在**测试时需快速推断隐状态**的场合尤其有用。下面给出一些例子。

27.7.3.1 语言建模与文档检索

我们可以用**类别型 RBM**给词袋 (bag-of-words) 定义生成模型，作为 LDA 的替代。一个微妙之处在于：无向模型中的配分函数依赖图的大小，因此依赖**文档长度**。Salakhutdinov & Hinton (2010) 提出了解法：使用**共享权重**的类别型 RBM，但把隐层激活偏置 c_k 乘以文档长度 L ，来补偿词频向量 v 随 L 变大这一事实：

$$E(v, h; \theta) = - \sum_{k,c} v_c h_k W_k^c - \sum_c v_c b_c - L \sum_k h_k c_k \quad (27.129)$$

其中 $v_c = \sum_{l=1}^L \mathbf{1}(y_{il} = c)$ 。这就好像只有一个**多项式结点**（因此去掉了下标 r ），它有 C 个状态（词表大小）。该模型称为**replicated softmax** (Salakhutdinov & Hinton, 2010)，是 mPCA/LDA 的**无向替代**。

我们可用**困惑度**在测试集上比较 RBM 与 LDA 的建模能力，困惑度可用**退火重要性采样 (AIS)** (§24.6.2) 估计。结果见图 27.32：LDA 明显优于统一词频 (unigram) 模型，而 RBM 又明显优于 LDA。

(原文此处有一处笔误：写成 "Another advantage of the LDA is that inference is fast and exact... as in Eq 27.107"，但式 (27.107) 来自RBM，且RBM的 $p(h | v)$ 是并行、快速、精确的；LDA 的推断一般要用变分或采样近似。因此应为 "Another advantage of the RBM..."。)

除了更准，**RBM 的推断也更快**：一次矩阵-向量乘 + sigmoid，如式 (27.107)。图 27.33 展示了在两个语料上的**精确率-召回率曲线**：取测试集一篇作为查询，计算其与所有训练文档的**主题向量余弦相似度**，对不同的返回数 M 画出曲线；若检索到的文档与查询**类别标签**相同，则视为相关（标签只用于评估）。

27.7.3.2 用于协同过滤的 RBM

RBM 已用于 Netflix 协同过滤竞赛 (Salakhutdinov et al., 2007)。事实上，**二值隐 + 类别可见的 RBM 略优于 SVD**。两者结合会更好。（最终冠军是多种模型的集成；Koren, 2009a。）

通俗解释 (整节 27.7)

- RBM 是无向版的“两层编码器-解码器”**：隐层与可见层**双向连接**、**层内无边**，因此 $p(h | v)$ 完全**可分解**，像前馈网络一样一次前向就能算出**隐层激活**。
- PoE vs 混合**：PoE 是把多个“规则/约束”**相乘**，能得到**更尖锐**的分布；混合是**加权相加**，往往更“软”。
- 分布式编码**：不像“祖母细胞”（一次只开一个主题），RBM 允许多个**隐单元同时激活**来表达复杂概念。

- **多种变体**: 二值/类别/高斯; 还可把隐/见设为高斯 (但那会退化成等价的有向因子分析或弱于二值隐)。
 - **训练难点**: 目标里有配分函数 Z , 直接求期望很难; **CD (对比散度) **用“从数据出发的短链 Gibbs”近似负相项; PCD维护跨迭代的持久链, 通常更稳。
 - **应用**:
 - **文本**: replicated softmax (长度补偿) → 在困惑度与检索上常优于 LDA, 且推断更快。
 - **协同过滤**: RBM 可与矩阵分解互补, 组合更强。
-

要点小结 (记忆版)

- **定义**: RBM = 无向两层图 (层内无边); 联合分布 $\propto e^{-E(v,h)}$ 。
- **推断**: $p(h \mid v)$ 条件独立、一次前向即可 (sigmoid); 生成 $p(v \mid h)$ 对称。
- **变体**:
 - 二值 RBM: 最常用, 式 (27.101–104)。
 - 类别 RBM: 1-of-C, 式 (27.105–107)。
 - 高斯 RBM: 实值数据, 式 (27.108–111)。
- **学习**: 梯度 = (数据期望) – (模型期望); 后者用Gibbs/CD/PCD近似; 配合 SGD + weight decay 等技巧。
- **直觉**: PoE 把多个“约束/主题”乘起来 → 强表达力、分布尖锐; RBM 隐层是**分布式多主题**。
- **应用亮点**:
 - 文本 (replicated softmax) : 困惑度/检索优于 LDA, 且推断快。
 - 协同过滤: RBM 可略胜 SVD, 与其集成更强。

如果你想, 我可以用一个小玩具数据 (几维二值输入) 演示一次 CD-1 的权重更新, 让“看到一次数据→重构→更新”的流程变得直观。

这一节主要在讲什么? 我不太懂

已思考 9s

当然可以! 这节 (27.7) 主要讲的是**受限玻尔兹曼机 (RBM)** ——一种用**无向图**表示的两层生成模型, 用来学习数据里的“隐含特征/主题”。它和前面那些**有向**的潜变量模型

(LDA、mPCA 等) 不太一样。

用大白话说清楚

- **RBM 是什么?**

就像一个两层“开关网络”：下面一层是**可见层**（数据），上面一层是**隐层**（很多二值开关）。**层内没有连线**，只有层与层之间连线。

这让**推断很快**：给定数据 v ，每个隐开关 h_k 都**彼此独立**，一次矩阵乘 + Sigmoid 就能得到它们的激活概率（并行算，像前馈神经网的前向）。

- **为什么不是“混合”而是“乘积”？(PoE 思想)**

RBM 属于“**专家乘积**”模型：多个“专家”（每条边的势函数）**相乘**来刻画数据，每个专家像一个**约束**（开/关/不关心）。相乘会让分布更“**尖**”（更有判别力），比“加权相加”（混合模型）更能表达复杂共现。

- 和“只开一个”的混合模型不同

混合模型像“**祖母细胞**”：一次只激活一个隐变量；RBM 是“**分布式编码**”：一次能**同时**激活多个隐单元来组合表达复杂模式（比如一篇文章同时是“政府+黑帮+花花公子”主题）。

- **变体有哪些？**

- **二值 RBM**：最常见，输入/隐层都是 0/1。
- **类别 RBM**：输入是 1-of-C（处理词或离散标签）。
- **高斯 RBM**：输入是**实数**（图像像素等）；还可让方差也受隐层控制。
- 把隐层设成高斯一般**不如**二值隐层有表达力（而且会退化成跟有向因子分析等价）。

- **训练难点与招数**

目标里有**配分函数** Z ，直接最大似然的“模型期望”难算 →

- **Gibbs 采样**：块采样 $h \rightarrow v \rightarrow h \rightarrow \dots$ ；但烧入慢。
- **对比散度 (CD-k)**：每次从**真实样本**出发，只跑**几步** Gibbs，就拿重构样本近似“模型期望”。快、好用。
- **PCD (持续 CD) /SML**：维护**持续的马氏链**跨迭代更新，往往更稳。
- 实作上配**SGD、小批量、动量、L2（权重衰减）**等常规技巧。

- **能干嘛？**

- **文本 (词袋)**：用“**replicated softmax**”版本（把隐层偏置按文档长度缩放）做语言建模/检索。实验里常**优于 LDA**的困惑度，且**推断更快**（一次前向）。
- **协同过滤 (推荐)**：用二值隐+类别可见做评分建模，效果可**略胜** SVD；与矩阵分解等**集成**表现更好（Netflix 夺冠方案都是大集成）。

一个形象的心智模型

把 RBM 想成带随机开关的“**自编码器**”：

看一眼数据 → 点亮一串隐开关 → 再用这些开关去**重构**数据。

CD 的更新，就是“让原样本与重构样本更像原样本、远离模型当前会生成的样子”。

什么时候考虑 RBM？

- 需要**快速推断隐表示**（测试时只要一跳前向）。
- 数据里存在**多种并发特征需要“乘法式组合”**（而非谁占比高就谁上）。
- 做文本/推荐等任务，想要一个**无向、分布式、表达力强**的生成基块，或作为深度模型的一层。

要点小结

- RBM = **无向、两层、层内无边**；隐层给定数据时**条件独立**，推断快。
- 属于**专家乘积 (PoE)**，能把多个“约束/主题”**乘起来**，表达更尖锐。
- **变体**：二值 / 类别 (1-of-C) / 高斯输入等。
- **学习**：梯度 = 数据期望 - 模型期望；用 CD/PCD 近似，配合 SGD+正则。
- **应用**：文本 (replicated softmax, 常优于 LDA, 且快) 、协同过滤 (与 SVD 互补，集成更强)。

如果你想，我也可以用一个极小的二值示例，演示一次 CD-1 的“看→重构→更新”过程，直观感受 RBM 在学什么。

基于本对话里我贴过的整章原文，输出：
①知识点总结
(分点+关键词+用途) ②子章节关联 (依赖/对比/递进+关系句，附Mermaid可选) ③本章总结。

已思考 32s

下面基于你贴过的整章原文（第 27 章：离散数据的潜在变量模型）给出三部分输出。

① 知识点总结 (分点 + 关键词 + 用途)

- **数据表示与任务 (27.1)**
 - **关键词**：词袋 (bag-of-words) 、计数向量、稀疏矩阵、多通道特征、图/关系张量

- **用途：**把文本、问卷、评分、网络链接等离散数据统一成“分类/计数/二值”形式，便于建模 $p(y)$ 或 $p(n)$ 。
- **混合模型 Mixture (27.2.1)**
 - **关键词：**单一隐变量 q_i (主题/簇) 、Cat/Multinomial/Poisson
 - **用途：**快速做“每个文档属于一个主题”的粗聚类或朴素生成；可用于简单文本建模或多输出的无监督 Naive Bayes。
- **指数族 PCA / LSA (27.2.2)**
 - **关键词：**ePCA、catPCA (softmax 回归式生成) 、Poisson/Multinomial 似然、连续隐变量 z
 - **用途：**把计数/分类数据做“低维实向量”表示 (类似 PCA)，做主题/标签的连续嵌入；适合可解释性一般但表达灵活的降维。
- **mPCA / LDA / PLSI (27.2.3 & 27.3)**
 - **关键词：**LDA (Dirichlet–Multinomial) 、文档主题分布 π_i (在单纯形上) 、词-题矩阵 B 、**混合成员 (admixture)**
 - **用途：**文本主题发现、歧义词消歧、检索、摘要的主题空间表征；可计算困惑度评估；有 Gibbs / 变分 / 在线变分等高效推断。
- **Gamma–Poisson (GaP) 与 NMF (27.2.4)**
 - **关键词：**非负隐变量 z_i^+ 、Ga–Poi 共轭、**条件于总词数 \Rightarrow 等价 mPCA、NMF (退化 EM)**
 - **用途：**非负分解以得到稀疏部件 (topics/部件-基)，用于可解释的计数数据分解；可加稀疏先验 (Spike-and-Gamma)。
- **LDA 扩展 (27.4)**
 - **CTM (相关主题) :** logistic-normal 先验 \Rightarrow 主题相关图；可视化主题共现。
 - **DTM (动态主题) :** 主题随时间平滑演化；时间对齐检索、历史话题分析。
 - **LDA-HMM:** 词序 + 语法 (HMM) \times 语义 (LDA) ；无监督区分功能词/内容词。
 - **有监督/带输入的 LDA:** sLDA、判别式 LDA、DMR-LDA、labeled/partially labeled LDA、随机效应 MoE；
 - **用途：**让主题“更像现实” (相关/演化/语法) 并可做预测 (情感、标签、多标签)。
- **图结构 (27.5)**
 - **SBM:** 块-块连边率 $\eta_{a,b}$ ；**按连接偏好而非相互连接度聚类。**
 - **MMSBM:** 每节点混合成员 π_i ；边级角色对 $q_{i \rightarrow j}, q_{i \leftarrow j}$ 。
 - **RTM:** **文本 + 链接联合** (用经验主题 $q_i \otimes q_j$ 预测边)。

- **用途：**社群/角色挖掘、补边预测、带文本的网络建模。
 - **关系数据 (27.6)**
 - **IRM (无限关系) :** 多类型对象、多关系的 SBM 非参版 (DP) , 可学本体、联合聚类。
 - **PMF (协同过滤) :** 用户/物品低维向量内积 + 偏置 + 正则 + 时间；处理 MNAR/隐式反馈/曝光日志、多项选择。
 - **用途：**知识图谱类关系学习、推荐系统评分/点击/选择预测。
 - **RBM (受限玻尔兹曼机, 27.7)**
 - **关键词：**无向二层、层内无边、PoE、二值/类别/高斯可见, CD / PCD 训练、replicated softmax
 - **用途：**快速后验推断的生成块；文本建模（常优于 LDA 的困惑度）、与矩阵分解组合做推荐。
-

② 子章节关联 (依赖 / 对比 / 递进 + 关系句)

关系句 (简要版)

- 27.1 → 27.2：由数据表示出发，进入“如何用隐变量压缩/解释离散数据”的**方法族综述**。
- Mixture ↔ ePCA：Mixture 是**单一主题**、可解释但刚性；ePCA 是**连续隐向量**、表达力强但推断难。
- ePCA → mPCA/LDA：把 ePCA 的**自然参数换成概率单纯形 (对偶参数)**，得到 mPCA/LDA，增强可解释与稀疏性。
- GaP ↔ mPCA/NMF：不固定总词数的 GaP 在**条件总和**时退化成 mPCA；去先验的极限对应 NMF。
- 27.3 LDA (细化)：在 27.2 框架内对 LDA 给出**生成、推断、评估、在线学习**等实用细节。
- 27.4 (扩展)：在 LDA 基础上，引入**相关性 (CTM) 、时间 (DTM) 、语法 (LDA-HMM) 与监督/输入**，面向真实场景。
- 27.5 (图) → 27.6 (关系)：从**单类型图 (SBM/MMSBM/RTM)** 推广到**多类型、多关系 (IRM/PMF)**，把“主题/块”思想推广至网络/推荐。
- 27.7 (RBM)：与前述**有向模型**并列的**无向**替代；在文本/推荐上可与 LDA/PMF 形成**互补或集成**。

对比要点

- 局部式 vs 分布式编码:** Mixture (单一主题/祖母细胞) vs LDA/ePCA/RBM (多主题并存)。
- 有向 vs 无向:** 有向 (LDA/PMF) **训练易、推断慢/近似**; 无向 (RBM) **推断快、训练难 (Z)**。
- 自然参数 vs 对偶参数:** ePCA (logit 空间) vs mPCA/LDA (概率单纯形)。
- 文本 vs 图/关系/推荐:** 主题思想跨域复用: 文本 (LDA/CTM/DTM/RTM/RBM) → 图 (SBM/MMSBM) → 关系与推荐 (IRM/PMF)。

递进脉络 (Mermaid 可选)

```

graph TD
    A[27.1 数据表示] --> B[27.2 方法总览]
    B --> B1[Mixture]
    B --> B2[ePCA/LSA]
    B --> B3[mPCA/LDA]
    B --> B4[GaP/NMF]
    B3 --> C[27.3 LDA 细化]
    C --> C1[评估: 困惑度]
    C --> C2[推断: Gibbs/变分/在线]
    C --> C3[选K]
    C --> D[27.4 扩展]
    D --> D1[CTM 相关主题]
    D --> D2[DTM 动态主题]
    D --> D3[LDA-HMM 语法+语义]
    D --> D4[有监督/带输入 LDA]
    D --> D5[判别式 catPCA]
    B --> E[27.5 图: SBM/MMSBM/RTM]
    E --> F[27.6 关系: IRM/PMF]
    B --> G[27.7 RBM/replicated softmax]
    G --- C[与LDA对比/互补]
    G --- F[与PMF集成于推荐]
  
```

③ 本章总结 (给决策者/实操者看的版本)

- 核心问题:** 离散数据 (词、分类、计数、链接、关系) 如何用**低维隐结构**来解释与预测?
- 方法谱系:**
 - 单一簇** (Mixture) → 2) **连续隐向量** (ePCA/LSA) → 3) **概率单纯形的混合成员** (mPCA/LDA) → 4) **非负与计数共轭** (GaP/NMF)。

在此基础上, LDA成为标准工作马, 并衍生出**相关/动态/语法/监督/带输入等扩展**;

同时把“主题/块/嵌入”的思想推广到**图/关系/推荐** (SBM/MMSBM/IRM/RTM/PMF);

并提供一个**无向替代** (RBM/replicated softmax), 在**推断速度和表达力**上与有向模型互补。
- 推断与学习:**
 - 有向:** EM/变分/Gibbs/在线变分, 工程上成熟;
 - 无向:** 需近似“模型期望” (Gibbs、CD/PCD), 但测试时后验很快。
- 评估与选择:**
 - 文本建模看**困惑度**、检索 PR;
 - 图/关系看**社群/角色可解释与补边精度**;

- 推荐看RMSE/CTR/Top-N，重视偏置、时间、MNAR/隐式反馈与多项选择。
- 落地指南（简）：
 - **无序文本主题**: LDA 起步；要相关/时间/语法/监督 → CTM/DTM/LDA-HMM/sLDA/DMR。
 - **网络社群/角色/补边**: SBM/MMSBM；带文本 → RTM。
 - **多类型关系/本体**: IRM。
 - **推荐**: PMF (内积+偏置+时间+侧信息)；可与 RBM/回归/树等集成。
 - **推断速度优先且可接受无向训练复杂度**: 试 RBM (replicated softmax 用于文本)。

一句话：本章提供了一套从混合→连续→单纯形再到图/关系/无向的通用工具箱，把“**隐结构**”这件事在离散世界里做深做透；实践中按**数据形态与业务目标**选型，再用合适的**推断/正则/在线**技巧把它跑好。

请用通俗易懂的话总结本章，使我能够理解

已思考 8s

好的！这章（第 27 章）讲的是：**怎么用“潜在（看不见的）结构”来理解和预测各种离散数据**——比如文本里的词、问卷里的选项、计数、社交网络的连边、用户给电影的评分等。

想象你在整理一座巨大而杂乱的资料馆：

- 书=文档里的词；
- 书架=主题或社群；
- 借阅记录=谁和谁有联系、谁给谁打分。

本章教你用几套“整理法”，把表面杂乱的记录，压缩成**低维、可解释**的隐结构，进而**做理解**（发现主题/社群/本体）和**预测**（补词、补边、推荐）。

1) 统一视角：离散数据怎么表示

- **词袋/计数向量**: 不看词序，只统计每个词出现多少次（得到很稀疏的矩阵）。
- **多通道**: 不止文字，还能有“视觉词”“问卷题目”等并列的分类变量。
- **图/关系**: 人-人、人-物、甚至三元关系（蛋白 i 与 j 在化学 k 下是否相互作用）。都能化成**二值/计数的矩阵或张量**。

2) 三条主线的“整理法”（模型家族）

A. 文本与计数的“主题/部件”视角

- **混合模型**（每篇只属一个主题）
易懂但僵硬，像“给每本书只贴一个标签”。
- **连续隐向量**（ePCA/LSA）
把文档压到低维实数向量；表达力强，但解释性一般。
- **LDA（核心）**
每篇文档是**多个主题的配比**（混合成员），每个主题是一组词的分布——像“书同时放在多个相关书架上”。
 - **训练**：Gibbs 采样、变分推断、**在线变分**（适合大语料）。
 - **评估**：**困惑度**（越低越好）。
 - **扩展**：
 - **CTM**：主题之间相关（“商业”常伴随“金融”）。
 - **DTM**：主题随时间变化（1900s 的“神经科学”和 2000s 不同）。
 - **LDA-HMM**：把“语法”交给 HMM，“语义”交给 LDA。
 - **有监督/带输入**：让主题帮你做分类、打标签，或利用外部特征。
- **GaP / NMF**（非负分解）
强调“部件组合”，可得**稀疏、可解释**的部件/主题；与 LDA在思想上相近。

B. 图与多关系的“社群/角色/本体”视角

- **SBM**（随机分块）
按“连接偏好”给节点分组：A 组爱连 B 组，B 组爱连 C 组……不要求同组内部一定相连。
- **MMSBM**（混合成员 SBM）
节点可**身兼多职**：对不同对象扮演不同角色，能发现“跨圈子的人”。
- **RTM**（关系主题模型）
文本 + 链接一起学：两篇论文主题越像，越可能互相引用。
- **IRM**（无限关系模型）
多类型对象、多种关系的非参版“分块”，能自动学出**本体**（概念簇、谓词簇）。
- **PMF**（概率矩阵分解）
推荐系统主力：把用户/物品映射到同一低维空间，**内积≈评分**；再加**用户/物品偏置、时间变化、侧信息、隐式反馈/曝光日志**等，效果显著。

C. 无向能量模型的“专家乘积”视角

- **RBM (受限玻尔兹曼机)**

两层、层内无边，**多个“专家”相乘**来刻画数据；测试时**推断很快**（一次前向）。

- 变体：二值、类别（replicated softmax 适配词袋）、高斯。
 - 训练：CD/PCD（对比散度/持续对比散度）+ SGD。
 - 用途：文本建模（常能优于 LDA 的困惑度）、与矩阵分解**集成**做推荐。
-

3) 怎么选？一句话指南

- **无序文本做主题**：先用 LDA；若要主题相关/随时间/兼顾语法或做分类 → 用 CTM/DTM/LDA-HMM/**有监督 LDA**。
 - **找社群/角色/补边（图）**：SBM/MMSBM；有文本一起学 → RTM。
 - **多类型对象、多关系、本体发现**：IRM。
 - **推荐（评分/点击/选择）**：PMF（内积+偏置+时间+侧信息+隐式反馈/曝光日志）；也可和 RBM 等模型**组合**。
 - **推断速度优先且能接受无向训练复杂度**：试 RBM（文本用 replicated softmax）。
-

4) 训练与评估的常识性要点

- **推断**：有向（LDA/PMF）→ 变分/采样/在线较成熟；无向（RBM）→ 训练难点在配分函数，用 CD/PCD。
 - **评估**：文本看**困惑度/检索**；图看**社群解释/补边准确**；推荐看 RMSE / CTR / Top-N。
 - **规模化**：用**在线变分**（LDA）、SGD/**小批量/动量**（PMF/RBM）。
-

5) 把整章压成 8 句话

1. 离散数据都能变成**二值/计数/类别**的稀疏矩阵或张量。
2. 我们用**潜在结构**把它们压到低维：主题、社群、角色、因子。
3. LDA是文本主题发现的工作马：文档=多主题混合。
4. LDA 可扩展出**相关/动态/语法/有监督/带输入**等现实增强版。
5. SBM/MMSBM/RTM/IRM把“主题/块”的想法推广到**图与多关系**，能找社群、学本体、补边。
6. PMF是推荐系统基石：用户×物品=低维向量内积+偏置+时间+侧信息。
7. RBM用无向“专家乘积”建模，推断快，可做文本和推荐的强力基块或拼装件。

8. 真正落地看任务：**理解**（主题/社群/本体） vs **预测**（困惑度/补边/评分），选择合适家族与训练策略。
-

如果你告诉我你的**具体数据形态与目标**（比如“商品-用户点击”“科研论文与引用”“多语言文本分类”），我可以直接帮你把上面的选择落到一个**最小可用方案**上：选模型、选训练法、选评估指标。