

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。

Figure 11.1 A DGM with and without hidden variables. The leaves represent medical symptoms. The roots represent primary causes, such as smoking, diet and exercise. The hidden variable can represent mediating factors, such as heart disease, which might not be directly visible.

Figure 11.2 A latent variable model represented as a DGM.
 (a) Many-to-many. (b) One-to-many. (c) Many-to-one. (d) One-to-one.

Table 11.1 Summary of some popular directed latent variable models. Here “Prod” means product, so “Prod. Discrete” in the likelihood means a factored distribution of the form $\prod_j \text{Cat}(x_{ij} | z_i)$, and “Prod. Gaussian” means a factored distribution of the form $\prod_j N(x_{ij} | z_i)$. “PCA” stands for “principal components analysis”. “ICA” stands for “independent components analysis”.

11 Mixture models and the EM algorithm

11.1 Latent variable models

In Chapter 10 we showed how graphical models can be used to define high-dimensional joint probability distributions. The basic idea is to model dependence between two variables by adding an edge between them in the graph. (Technically the graph represents conditional independence, but you get the point.) An alternative approach is to assume that the observed variables are correlated because they arise from a hidden common “cause”. Model with hidden variables are also known as latent variable models or LVMs. As we will see in this chapter, such models are harder to fit than models with no latent variables. However, they can have significant advantages, for two main reasons. First, LVMs often have fewer parameters than models that directly represent correlation in the visible space. This is illustrated in Figure

11.1. If all nodes (including H) are binary and all CPDs are tabular, the model on the left has 17 free parameters, whereas the model on the right has 59 free parameters. Second, the hidden variables in an LVM can serve as a bottleneck, which computes a compressed representation of the data. This forms the basis of unsupervised learning, as we will see. Figure 11.2 illustrates some generic LVM structures that can be used for this purpose. In general there are L latent variables, z_1, \dots, z_L , and D visible variables, x_1, \dots, x_D , where usually $D > L$. If we have $L > 1$, there are many latent factors contributing to each observation, so we have a many-to-many mapping. If $L = 1$, we have only a single latent variable; in this case, z_1 is usually discrete, and we have a one-to-many mapping. We can also have a many-to-one mapping, representing different competing factors or causes for each observed variable; such models form the basis of probabilistic matrix factorization, discussed in Section 27.6.2. Finally, we can have a one-to-one mapping, which can be represented as $z_i \rightarrow x_i$. By allowing z_i and/or x_i to be vector-valued, this representation can subsume all the others. Depending on the form of the likelihood $p(x_i|z_i)$ and the prior $p(z_i)$, we can generate a variety of different models, as summarized in Table 11.1.

已思考 56s

下面先给出“逐字不遗漏”的中文翻译，然后用通俗比喻把要点讲清，最后做一个小结。

翻译

图 11.1 带有（左）和不带有（右）隐藏变量的有向图模型（DGM）。树叶结点表示医疗症状。根结点表示主要原因，比如吸烟、饮食和运动。隐藏变量可以表示“中介因素”，例如心脏病，这种因素可能不能被直接看见。

图 11.2 将潜变量模型表示成 DGM。（a）多对多。（b）一对多。（c）多对一。（d）一对一。

表 11.1 若干常见“有向潜变量模型”的汇总。“Prod”表示乘积，因此，在似然项里，“Prod. Discrete”表示形如 $\prod_j \text{Cat}(x_{ij} | z_i)$ 的分解（因子化）分布，而“Prod. Gaussian”表示形如

$\prod_j N(x_{ij} | z_i)$ 的分解分布。“PCA”指“主成分分析 (principal components analysis) ”。“ICA”指“独立成分分析 (independent components analysis) ”。

11 混合模型与 EM 算法

11.1 潜变量模型

在第 10 章中，我们展示了如何用图模型来定义高维的联合概率分布。基本思路是：如果两个变量相互依赖，就在图里给它们之间加一条有向边。（严格地说，图表达的是条件独立结构，但你能明白这层意思。）另一种做法是假设“观测到的变量之所以相关，是因为它们来自某个共同但隐藏的‘原因’”。带有隐藏变量的模型也称为**潜变量模型** (latent variable models, LVMs)。正如本章将看到的，这类模型比没有潜变量的模型更难拟合。不过它们有显著优势，主要有两个原因。

第一，LVM 往往比“直接在可见空间里刻画相关性”的模型参数更少。图 11.1 就说明了这一点。如果所有结点（包括 H ）都是二值的，且所有条件概率表 (CPD) 都是表格形式，那么左图的模型有 17 个自由参数，而右图有 59 个自由参数。

第二，LVM 里的隐藏变量可以充当一个**瓶颈**，它会对数据计算出一种压缩表示。这构成了无监督学习的基础，我们随后会看到。图 11.2 展示了几种可用于此目的的一般 LVM 结构。一般地，存在 L 个潜变量 z_{i1}, \dots, z_{iL} ，以及 D 个可见变量 x_{i1}, \dots, x_{iD} ，通常 $D \gg L$ 。如果 $L > 1$ ，每个观测由多个潜在因素共同贡献，于是得到**多对多映射**。如果 $L = 1$ ，我们只有一个潜变量；在这种情况下， z_i 通常是离散的，对应**一对多映射**。我们也可以有**多对一映射**，表示对每个观测变量都有不同的竞争性因素或原因；这种模型构成了**概率矩阵分解**的基础（见 27.6.2 节）。最后，我们还可以有一对一映射，可写作 $z_i \rightarrow x_i$ 。如果允许 z_i 和/或 x_i 取向量值，这种表示可以包含上面所有其它情况。根据似然 $p(x_i | z_i)$ 与先验 $p(z_i)$ 的具体形式，我们可以得到许多不同模型，表 11.1 做了汇总。

解释（通俗版）

1) 什么是 DGM、潜变量、和“隐藏的原因”？

- DGM（有向图模型）里，每个圆圈是一个随机变量，箭头表示“谁影响谁”。
- 观测到的变量（如体检里的多个指标）往往互相相关。与其在它们之间画很多边，不如假设这些相关性来自同一个（或多个）看不见的共同原因，把这个原因画成“隐藏变量”（例如“是否患有心脏病”）。这就是潜变量模型。

2) 为什么左图 17 个参数、右图 59 个参数？（一步步算给你看）

规则：一个二值结点若有 k 个父结点，它的 CPD 需要 2^k 个自由参数；没有父结点的二值

变量需要 1 个参数（相当于“为 1 的概率”）。

- **左图（有隐藏变量）**

- 顶部 3 个根结点：各 1 个参数 → 共 3。
- 隐藏结点 H ：有 3 个父结点 → 需要 $2^3 = 8$ 个。
- 底部 3 个症状：每个只有 H 1 个父结点 → 每个 $2^1 = 2$ ，共 $3 \times 2 = 6$ 。
- 合计 $3 + 8 + 6 = 17$ 。

- **右图（没有隐藏变量、让症状之间直接连边）**

- 顶部 3 个根结点：各 1 个 → 3。
- 左下症状：父结点是上面 3 个 → $2^3 = 8$ 。
- 中下症状：父结点是上面 3 个 + 左下 1 个 → $2^4 = 16$ 。
- 右下症状：父结点是上面 3 个 + 左下 + 中下 → $2^5 = 32$ 。
- 合计 $3 + 8 + 16 + 32 = 59$ 。

直观理解：用一个“病”这个隐藏变量，能把“吸烟/饮食/运动”等原因与“多个症状”之间错综复杂的相关性归并起来；参数大幅减少，模型更简洁。

3) “瓶颈”和压缩表示

隐藏变量把信息从高维的观测（很多体征）压到低维的潜在因子（少数“健康指数”）。这就像把一张高清图片压缩成几条主要特征（亮度、色调等），仍能复原大体内容。这样的压缩是无监督学习里常见的目标。

4) 图 11.2 的四种结构

- **多对多 (a)**：很多潜因子影响很多观测；每个观测也由多个因子共同决定。像“身高体重基因 + 饮食习惯 + 运动”共同影响多个体检指标。
- **一对多 (b)**：一个潜变量决定很多观测。典型是混合模型/聚类： z_i 选了“第几类”，然后该类生成整条样本。
- **多对一 (c)**：多个潜因素只生成一个观测值（对每个 x 都有一组因子）。这与矩阵分解（把评分拆成“用户因子 × 物品因子”）是一个思路。
- **一对一 (d)**：每个 x_i 对应一个 z_i 。如果允许 z_i 、 x_i 是向量，就能覆盖上面几种结构。

5) 表 11.1 怎么读？“Prod”到底是啥？

- “Prod”=“乘积/因子化”。意思是在给定潜变量的条件下，观测维度彼此独立，整体分布是“各维度分布的乘积”。
 - 例如 “Prod. Discrete” = $\prod_j \text{Cat}(x_{ij} | z_i)$ ：每个离散维度各自服从一个分类分布；给定 z_i 后互不影响。

- “Prod. Gaussian” = $\prod_j N(x_{ij} | z_i)$: 每个维度条件高斯，彼此独立。
- 常见组合与直觉：
 - **高斯混合 (Mixture of Gaussians)** : $p(z_i)$ 离散、 $p(x_i | z_i)$ 多元高斯 (MVN) 。适合连续数据的“聚类”。
 - **多项式混合**: $p(z_i)$ 离散、 $p(x_i | z_i)$ Prod. Discrete。适合“词袋”这类离散计数数据的聚类。
 - **因子分析/概率 PCA**: $p(z_i)$ 是独立高斯、 $p(x_i | z_i)$ 是 Prod. Gaussian (条件独立)。做降维与连续特征提炼。
 - **概率 ICA/稀疏编码**: $p(z_i)$ 用拉普拉斯 (尖峰厚尾) \rightarrow 鼓励稀疏因子； $p(x_i | z_i)$ Prod. Gaussian。
 - **Multinomial PCA**: 观测是离散 (如词)，潜因子是高斯。
 - **LDA (潜狄利克雷分配)** : $p(x_i | z_i)$ Prod. Discrete； $p(z_i)$ 狄利克雷。做文档主题发现。
 - **BN20/QMR**: Noisy-OR 似然 + 伯努利先验，常用于“多种疾病 (伯努利) \rightarrow 多种症状 (Noisy-OR) ”。
 - **Sigmoid 信念网**: 伯努利 \rightarrow 伯努利，用 sigmoid 连接，是深度生成模型的早期形式。

术语小抄：

Cat=Categorical (分类分布)；Bernoulli (二项 0/1)；Dirichlet (狄利克雷，常做“概率向量”的先验)；Laplace (拉普拉斯，稀疏友好)；MVN=多元高斯；Noisy-OR (“多个原因各自以一定概率触发结果”的逻辑门)。

6) 为什么这些模型“更难拟合”？

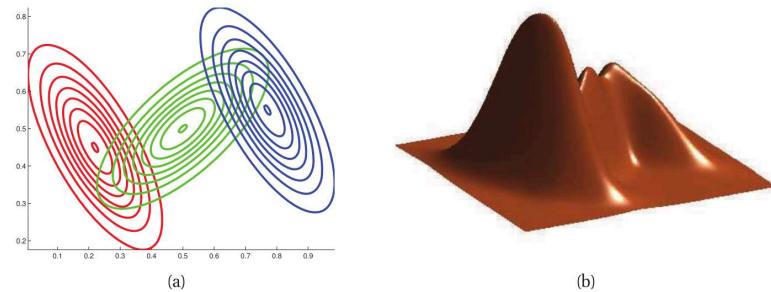
因为潜变量看不见，不能直接把似然写成已知数据的函数来最大化。常用办法是 EM 算法：

- **E 步 (推断)**：在当前参数下，估计/计算潜变量的分布 (谁属于哪个簇、因子取多大)。
- **M 步 (学习)**：用 E 步得到的“软赋值”来更新参数。
两步交替，直到收敛。

要点小结

- **潜变量模型 (LVM) **用“隐藏的共同原因”解释观测到的相关性；它是有向图模型的一种常见用法。
- **优势**：

1. **参数更少** (图 11.1: 17 vs 59, 计算如上)。
 2. **形成瓶颈/压缩表示**, 是无监督学习与降维的基础。
- **结构形态** (图 11.2) : 多对多、一对多 (混合/聚类) 、多对一 (矩阵分解) 、一对一 (向量化可涵盖全部)。
 - “Prod.”=**条件独立**: 给定潜变量, 各维观测相互独立 → 整体是各维分布的乘积。
 - **表 11.1**列出了常见选择 (高斯混合、离散混合、因子分析/PPCA、ICA/稀疏编码、Multinomial PCA、LDA、Noisy-OR 疾病-症状模型、Sigmoid 信念网), 它们由似然 $p(x | z)$ 与先验 $p(z)$ 的不同搭配构成。
 - **训练**: 因潜变量不可见, 通常用 EM 算法在“推断潜变量—更新参数”之间迭代。



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。

Figure 11.3 A mixture of 3 Gaussians in 2d. (a) We show the contours of constant probability for each component in the mixture. (b) A surface plot of the overall density. Based on Figure 2.23 of (Bishop 2006a). Figure generated by mixGaussPlotDemo.

11.2 Mixture models

The simplest form of LVM is when $z_i \in \{1, \dots, K\}$, representing a discrete latent state. We will use a discrete prior for this, $p(z_i) = \text{Cat}(\pi)$. For the likelihood, we use $p(x_i | z_i = k) = p_k(x_i)$, where p_k is the k 'th base distribution for the observations; this can be of any type. The overall model is known as a mixture model, since we are mixing together the K base distributions as follows:

$$p(x_i | \theta) = \sum_{k=1}^K \pi_k p_k(x_i | \theta) \quad (11.1)$$

This is a convex combination of the p_k 's, since we are taking a weighted sum, where the mixing weights π_k satisfy $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^K \pi_k = 1$. We give some

examples below.

11.2.1 Mixtures of Gaussians

The most widely used mixture model is the mixture of Gaussians (MOG), also called a Gaussian mixture model or GMM. In this model, each base distribution in the mixture is a multivariate Gaussian with mean μ_k and covariance matrix Σ_k . Thus the model has the form

$$p(x_i|\theta) = \sum_{k=1}^K \pi_k N(x_i|\mu_k, \Sigma_k) \quad (11.2)$$

Figure 11.3 shows a mixture of 3 Gaussians in 2D. Each mixture component is represented by a different set of elliptical contours. Given a sufficiently large number of mixture components, a GMM can be used to approximate any density defined on RD.

11.2.2 Mixture of multinoullis

We can use mixture models to define density models on many kinds of data. For example, suppose our data consist of D-dimensional bit vectors. In this case, an appropriate classconditional density is a product of Bernoullis:

$$p(x_i|z_i = k, \theta) = \prod_{j=1}^D \text{Ber}(x_{ij}|\mu_{jk}) = \prod_{j=1}^D \mu_{x_{ij}}^{z_{jk}} (1 - \mu_{jk})^{1-z_{jk}} \quad (11.3)$$

where μ_{jk} is the probability that bit j turns on in cluster k . The latent variables do not have to any meaning, we might simply introduce latent variables in order to make the model more powerful. For example, one can show (Exercise 11.8) that the mean and covariance of the mixture distribution are given by

$$E[x] = \sum_k \pi_k \mu_k \quad (11.4)$$

$$\text{cov}[x] = \sum_k \pi_k [\Sigma_k + \mu_k \mu_k^T] - E[x] E[x]^T \quad (11.5)$$

where $\Sigma_k = \text{diag}(\mu_{jk}(1 - \mu_{jk}))$. So although the component distributions are factorized, the joint distribution is not. Thus the mixture distribution can capture correlations between variables, unlike a single product-of-Bernoullis model.

已思考 7s

下面先逐字翻译原文（不遗漏），再用通俗解释，最后给出要点小结。

翻译

图 11.3 二元 (2D) 空间中的 3 个高斯的混合。(a) 展示了混合中每个分量的“等概率密度曲线”(等高线)。(b) 展示了整体密度的三维曲面图。基于 (Bishop 2006a) 图 2.23。图由 `mixGaussPlotDemo` 生成。

11.2 混合模型

最简单的潜变量模型是令 $z_i \in \{1, \dots, K\}$, 表示一个离散的潜在状态。我们为它使用一个离散先验 $p(z_i) = \text{Cat}(\boldsymbol{\pi})$ 。在似然项里, 我们用

$$p(\mathbf{x}_i \mid z_i = k) = p_k(\mathbf{x}_i),$$

其中 p_k 是第 k 个用于观测的“基 (底) 分布”; 它可以是任何类型。整个模型称为**混合模型**, 因为我们把 K 个基分布“混合”在一起:

$$p(\mathbf{x}_i \mid \theta) = \sum_{k=1}^K \pi_k p_k(\mathbf{x}_i \mid \theta) \quad (11.1)$$

这对 p_k 来说是一个**凸组合**, 因为我们在做加权求和, 且混合权重 π_k 满足 $0 \leq \pi_k \leq 1$ 且 $\sum_{k=1}^K \pi_k = 1$ 。下面给出若干例子。

11.2.1 高斯混合

使用最广泛的混合模型是**高斯混合模型** (Mixture of Gaussians, MOG, 也称 Gaussian Mixture Model, GMM)。在该模型中, 混合里的每个基分布都是一个多元高斯, 其均值为 $\boldsymbol{\mu}_k$, 协方差矩阵为 Σ_k 。因此模型形式为

$$p(\mathbf{x}_i \mid \theta) = \sum_{k=1}^K \pi_k N(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k) \quad (11.2)$$

图 11.3 展示了二维空间中 3 个高斯的混合。每个混合分量由一组椭圆形等高线表示。只要混合分量足够多, GMM 就可以逼近定义在 R^D 上的任意密度。

11.2.2 多伯努利混合 (Mixture of Multinoullis)

我们可以用混合模型来为多种数据定义密度模型。举例来说, 假设我们的数据是 D 维的比特向量。在这种情况下, 一个合适的“类条件”密度是**伯努利乘积**:

$$p(\mathbf{x}_i \mid z_i = k, \theta) = \prod_{j=1}^D \text{Ber}(x_{ij} \mid \mu_{jk}) = \prod_{j=1}^D \mu_{jk}^{x_{ij}} (1 - \mu_{jk})^{1-x_{ij}} \quad (11.3)$$

其中 μ_{jk} 是在第 k 个簇 (cluster) 中, 第 j 位比特取 1 的概率。潜变量本身不必有任何语义; 我们也可以仅仅为了让模型更强大而把它们引入。比如可以证明 (练习 11.8) 混合分布的均值与协方差为

$$E[\mathbf{x}] = \sum_k \pi_k \boldsymbol{\mu}_k \quad (11.4)$$

$$\text{cov}[\mathbf{x}] = \sum_k \pi_k [\boldsymbol{\Sigma}_k + \boldsymbol{\mu}_k \boldsymbol{\mu}_k^\top] - E[\mathbf{x}]E[\mathbf{x}]^\top \quad (11.5)$$

其中 $\boldsymbol{\Sigma}_k = \text{diag}(\mu_{jk}(1 - \mu_{jk}))$ 。因此，尽管**分量分布**是因子化（各维条件独立）的，**总体联合分布**却不是。于是，与单一的“伯努利乘积模型”不同，**混合分布能够刻画变量之间的相关性**。

通俗解释

1) 混合模型在干嘛？

想像你手里有几把“喷雾枪”，每把都有自己的形状：高斯就像一个“圆润的雾团”，伯努利乘积就像“每个维度独立开/关的开关板”。混合模型就是：先用一个“掷色子”步骤 ($z_i \sim \text{Cat}(\boldsymbol{\pi})$) 挑选一把喷雾枪，再用被选中的那把喷出一个样本 \mathbf{x}_i 。整体分布就是把所有枪喷出的雾混在一起——这就是式 (11.1) 的“加权求和”。

2) 为什么说是“凸组合”？

因为权重 π_k 都非负且加起来为 1，正如“平均值”是凸组合一样。它保证混合后的结果仍是一个合法的概率分布。

3) 高斯混合 (GMM)

- 每个分量是一个多元高斯 $N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ 。
- 图 11.3(a) 里每个椭圆是一条“等密度线”，三种颜色对应 3 个分量。
- 图 11.3(b) 把三者加起来后的整体密度画成了三维地形：会出现多个“山峰”（多模态）。
- 直觉：只要分量够多、位置和形状调得好，很多复杂的分布都能被“多个高斯山峰的和”逼近。

4) 多伯努利混合 (Mixture of Multinoullis) 是啥？

- 数据是 0/1 向量（例如黑白图像的像素、某人是否点击过每个商品）。
- 在**固定分量** k 内，每个维度独立，且第 j 维取 1 的概率是 μ_{jk} 。这就是“伯努利乘积”（各维度概率相乘）。
- 但是把不同分量混在一起后，整体上的各维就会**相关了**：因为“属于同一簇”的样本更可能一起出现某些模式（例如“这类文档经常同时包含词 A 和词 B”）。
- 公式 (11.4)–(11.5) 用“全期望/全方差定理”给出了混合分布的均值与协方差：
 - 均值是“各簇均值的加权平均”。

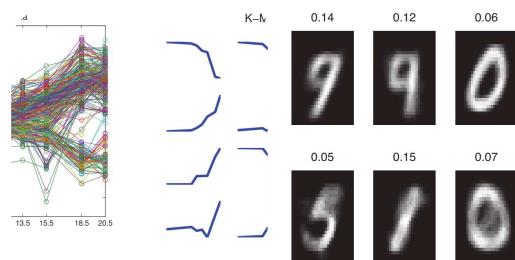
- 协方差 = “簇内方差的加权平均” + “不同簇的均值差异带来的方差” ($\mu_k \mu_k^\top$ 那项)
 - “总体均值的外积”。正是第二部分让总体出现相关性。
- 对于伯努利分量， Σ_k 是对角阵，元素 $\mu_{jk}(1 - \mu_{jk})$ 就是伯努利的方差。

5) 为什么要引入潜变量 z ?

- 让模型能自动“分群/分模式”：不同 k 代表不同“簇/模式”。
- 即使每个簇里假设维度独立（好算），混在一起后仍能表达复杂相关性（更强大）。
- 这就是混合模型既灵活（可表达多峰/相关）又可计算（分量内简单）的原因。

要点小结

- **混合模型**：先抽一个离散潜变量 $z_i \sim \text{Cat}(\pi)$ ，再从对应分量 $p_k(\mathbf{x})$ 生成数据；总体密度 $p(\mathbf{x}) = \sum_k \pi_k p_k(\mathbf{x})$ 是对分量的**凸组合**。
- **GMM/MOG**： p_k 取多元高斯。能用多个“高斯山峰”逼近任意连续密度；图 11.3 展示了 3 个分量的等高线与整体密度。
- **多伯努利混合**：数据为 0/1 向量；在簇内各维独立， $p(\mathbf{x} | z = k) = \prod_j \text{Ber}(x_j | \mu_{jk})$ 。
- **均值/协方差**： $E[\mathbf{x}] = \sum_k \pi_k \mu_k$ ； $\text{cov}[\mathbf{x}] = \sum_k \pi_k (\Sigma_k + \mu_k \mu_k^\top) - E[\mathbf{x}]E[\mathbf{x}]^\top$ 。尽管**分量内独立，但总体不独立**，因此能表达相关性。
- **实用价值**：自动发现隐藏簇/模式；在保持计算可控的同时获得更强的表达能力。



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。

Figure 11.4 (a) Some yeast gene expression data plotted as a time series. (c) Visualizing the 16 cluster centers produced by K-means. Figure generated by kmeansYeastDemo.

Figure 11.5 We fit a mixture of 10 Bernoullis to the binarized MNIST digit data. We show the MLE for the

corresponding cluster means, μ_k . The numbers on top of each image represent the mixing weights $\hat{\pi}_k$. No labels were used when training the model. Figure generated by mixBerMnistEM.

11.2.3 Using mixture models for clustering

There are two main applications of mixture models. The first is to use them as a black-box density model, $p(x_i)$. This can be useful for a variety of tasks, such as data compression, outlier detection, and creating generative classifiers, where we model each class-conditional density $p(x|y=c)$ by a mixture distribution (see Section 14.7.3). The second, and more common, application of mixture models is to use them for clustering. We discuss this topic in detail in Chapter 25, but the basic idea is simple. We first fit the mixture model, and then compute $p(z_i = k|x_i, \theta)$, which represents the posterior probability that point i belongs to cluster k . This is known as the responsibility of cluster k for point i , and can be computed using Bayes rule as follows:

$$r_{ik} = p(z_i = k|x_i, \theta) = \frac{p(z_i = k|\theta)p(x_i|z_i = k, \theta)}{\sum_{k=1}^K p(z_i = k|\theta)p(x_i|z_i = k, \theta)} \quad (11.6)$$

This procedure is called soft clustering, and is identical to the computations performed when using a generative classifier. The difference between the two models only arises at training time: in the mixture case, we never observe z_i , whereas with a generative classifier, we do observe y_i (which plays the role of z_i).

We can represent the amount of uncertainty in the cluster assignment by using $1 - \max_k r_{ik}$. Assuming this is small, it may be reasonable to compute a hard clustering using the MAP estimate, given by

$$z^* = \arg\max_k \log p(x_i|z_i = k, \theta) + \log p(z_i = k|\theta) \quad (11.7)$$

Hard clustering using a GMM is illustrated in Figure 1.8, where we cluster some data representing the height and weight of people. The colors represent the hard assignments. Note that the identity of the labels (colors) used is immaterial; we are free to rename all the clusters, without affecting the partitioning of the data; this is called label switching.

Another example is shown in Figure 11.4. Here the data vectors $x_i \in \mathbb{R}^7$ represent the expression levels of different

genes at 7 different time points. We clustered them using a GMM. We see that there are several kinds of genes, such as those whose expression level goes up monotonically over time (in response to a given stimulus), those whose expression level goes down monotonically, and those with more complex response patterns. We have clustered the series into $K = 16$ groups. (See Section 11.5 for details on how to choose K .) For example, we can represent each cluster by a prototype or centroid. This is shown in Figure 11.4(b). As an example of clustering binary data, consider a binarized version of the MNIST handwritten digit dataset (see Figure 1.5(a)), where we ignore the class labels. We can fit a mixture of Bernoullis to this, using $K = 10$, and then visualize the resulting centroids, μ^k , as shown in Figure 11.5. We see that the method correctly discovered some of the digit classes, but overall the results aren't great: it has created multiple clusters for some digits, and no clusters for others. There are several possible reasons for these "errors":

- The model is very simple and does not capture the relevant visual characteristics of a digit. For example, each pixel is treated independently, and there is no notion of shape or a stroke.
- Although we think there should be 10 clusters, some of the digits actually exhibit a fair degree of visual variety. For example, there are two ways of writing 7's (with and without the cross bar). Figure 1.5(a) illustrates some of the range in writing styles. Thus we need $K = 10$ clusters to adequately model this data. However, if we set K to be large, there is nothing in the model or algorithm preventing the extra clusters from being used to create multiple versions of the same digit, and indeed this is what happens. We can use model selection to prevent too many clusters from being chosen but what looks visually appealing and what makes a good density estimator may be quite different.
- The likelihood function is not convex, so we may be stuck in a local optimum, as we explain in Section 11.3.2. This example is typical of mixture modeling, and goes to show one must be very cautious trying to "interpret" any clusters that are discovered by the method. (Adding a little bit of supervision, or using informative priors, can help a lot.)

下面先把原文“完整不遗漏”地译成中文，然后用通俗的方式解释，最后给出要点小结。

翻译

图 11.4 (a) 把一些酵母基因的表达数据画成时间序列。(b) 可视化用 K-means 得到的 16 个簇中心。图由 `kmeansYeastDemo` 生成。

图 11.5 我们把 10 个伯努利分布的混合拟合到二值化后的 MNIST 手写数字数据上。图中显示的是相应簇均值 μ_k 的极大似然估计。每张图上方的数字表示混合权重 π_k 。训练模型时没有使用任何标签。图由 `mixBerMnistEM` 生成。

11.2.3 将混合模型用于聚类

混合模型有两个主要应用。第一个是把它们当作一个“黑盒”的密度模型 $p(x_i)$ 。这对多种任务都很有用，比如数据压缩、异常点检测、以及构建生成式分类器——在生成式分类器中，我们把每个类条件密度 $p(x | y = c)$ 建模为一个混合分布（见 14.7.3 节）。第二个、也是更常见的应用，是把混合模型用于聚类。我们在第 25 章会详细讨论这一主题，但基本思想很简单。先拟合混合模型，然后计算

$$p(z_i = k | x_i, \theta),$$

它表示第 i 个样本属于簇 k 的后验概率。这被称为“簇 k 对样本 i 的责任度 (responsibility)”，可以用贝叶斯公式计算如下：

$$r_{ik} = p(z_i = k | x_i, \theta) = \frac{p(z_i = k | \theta) p(x_i | z_i = k, \theta)}{\sum_{k=1}^K p(z_i = k | \theta) p(x_i | z_i = k, \theta)} \quad (11.6)$$

这种做法叫作**软聚类**，而且与使用生成式分类器时所做的计算完全相同。二者的区别只出现在训练阶段：在混合模型里，我们从未观测到 z_i ；而在生成式分类器里，我们观测到了 y_i （其角色与 z_i 类似）。

我们可以用 $1 - \max_k r_{ik}$ 来表示样本在簇分配上的不确定性。若该值很小，那么用 MAP 估计进行**硬聚类**是合理的，其形式为

$$z_i^{*} = \arg \max_k r_{ik} = \arg \max_k [\log p(x_i | z_i = k, \theta) + \log p(z_i = k | \theta)] \quad (11.7)$$

使用 GMM 进行硬聚类的一个例子见图 1.8，我们把一些表示人身高与体重的数据进行聚类。颜色代表硬分配。注意，所用标签（颜色）的“身份”无关紧要；我们可以把所有簇重命名而不影响数据的划分；这称为**标签置换** (label switching)。

另一个例子见图 11.4。这里的数据向量 $x_i \in \mathbb{R}^7$ 表示 7 个不同时间点上的基因表达水平。我们用 GMM 对它们做聚类。可以看到存在多种类型的基因：例如表达水平随时间单调上升（对某刺激的响应）、单调下降，或者具有更复杂的响应模式。我们把这些时间序列分成了 $K = 16$ 组。（关于如何选择 K 的细节见 11.5 节。）例如，我们可以用一个原型或质心来表示每个簇，如图 11.4(b) 所示。

再看一个对**二值数据**聚类的例子：把 MNIST 手写数字数据（见图 1.5(a)）二值化，并忽略类别标签。我们可以用 $K = 10$ 的伯努利混合去拟合它，然后把得到的质心 $\hat{\mu}_k$ 可视化，如图 11.5 所示。可以看到，方法确实找到了某些数字类别，但总体效果并不好：对某些数字它产生了多个簇，而对另一些数字又没有簇。造成这些“错误”的原因可能有以下几点：

- 该模型非常简单，未能捕捉数字的相关视觉特征。例如它把每个像素独立对待，并没有“形状”或“笔画”的概念。
- 虽然我们认为应该有 10 个簇，但某些数字实际上具有很强的外观多样性。例如数字 7 有两种写法（有横杠和没横杠）。图 1.5(a) 展示了书写风格的范围。因此，要充分建模这些数据需要 $K > 10$ 个簇。然而一旦把 K 设得很大，模型或算法并没有任何机制阻止“额外的簇”去为同一个数字创建多个版本，而这确实会发生。我们可以用模型选择来避免选择过多的簇，但“看起来好看”的可视化与“作为密度估计器表现好”这两件事可能完全不同。
- 似然函数不是凸的，因此我们可能会陷入**局部最优**（11.3.2 节会解释）。

这个例子是混合建模中的典型现象，它说明：对于由该方法“发现”的任何簇，在尝试“解释”它们时必须非常谨慎。（加入少量监督，或使用信息性先验，往往能帮助很多。）

通俗解释

1) 混合模型做聚类的流程

- 第一步：用训练数据把混合模型（如 GMM）拟合好：学出每个分量的参数和权重 π_k 。
- 第二步：对每个样本 x_i ，算它属于每个簇的概率 r_{ik} （式 11.6）。这就是“软分配”，每个样本对所有簇都有一个“归属度”。
- 如果想要**硬分配**，就选概率最大的那个簇（式 11.7）。式子里的对数一项来自于把分子取对数： $\log p(x | z = k) + \log p(z = k)$ 。

2) 责任度 r_{ik} 的直觉

分子 = “先抽到簇 k 的机会” \times “在簇 k 下生成 x_i 的可能性”；分母把所有簇都加起来做归一化，所以得到的是“在看到 x_i 后，属于簇 k 的后验概率”。

3) 软聚类 vs 硬聚类

- 软：保留不确定性（每个 r_{ik} 介于 0–1 且总和为 1）。
- 硬：直接做决策（取最大）。不确定性可以用 $1 - \max_k r_{ik}$ 衡量。

4) 为什么会“标签置换”？

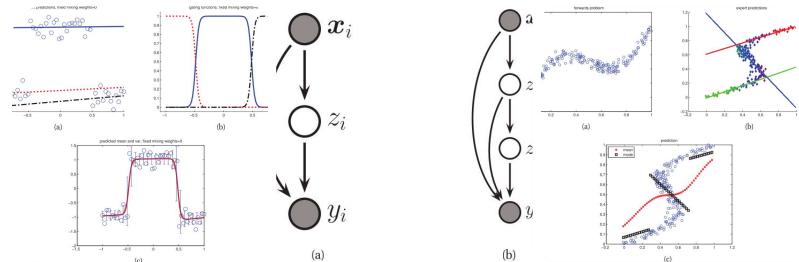
簇的编号本身没有意义，只是“第 1、2、...、K 个分量”的记号。如果我们把 1 和 2 互换，模型生成的数据分布不变，所以颜色/编号可以任意换名。

5) 两个例子

- **基因表达**：每个样本是 7 个时间点的曲线。GMM 能把“上升型/下降型/复杂型”等模式分到不同簇里；用 K-means 也能给出 16 个代表曲线（质心）。
 - **二值 MNIST**：用伯努利混合（每个像素独立的 0/1 概率）。能大致找出一些数字，但也会把同一个数字拆成多个簇、或漏掉一些数字。原因：模型太简单、数字风格多样导致需要更多分量、以及训练会陷局部最优。
-

要点小结

- 混合模型既可当作**密度模型**用于压缩/异常/生成式分类，也可用于**聚类**。
- **责任度** $r_{ik} = p(z_i = k | x_i, \theta)$ （式 11.6）给出软聚类；其最大值对应的簇给出硬聚类（式 11.7），不确定性用 $1 - \max_k r_{ik}$ 衡量。
- **标签置换**：簇的编号无意义，交换不影响划分。
- 例子显示：GMM 能把时间序列分成多种响应模式；伯努利混合在 MNIST 上效果有限。
- 差效果的常见原因：
 1. 分量模型过于简单（忽略形状/空间关系）；
 2. 真实数据的模式数 > 设定的 K 或者“额外分量”重复建模同一模式；
 3. 训练目标非凸 → 容易落入局部最优。
- 结论：用混合模型做聚类时，要谨慎解读簇；必要时加入少量监督或信息性先验，并用模型选择合理确定 K 。



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。

Figure 11.6 (a) Some data fit with three separate regression lines. (b) Gating functions for three different “experts”. (c) The conditionally weighted average of the three expert predictions. Figure generated by mixexpDemo.

Figure 11.7 (a) A mixture of experts. (b) A hierarchical mixture of experts.

Figure 11.8 (a) Some data from a simple forwards model. (b) Some data from the inverse model, fit with a mixture of 3 linear regressions. Training points are color coded by their responsibilities. (c) The predictive mean (red cross) and mode (black square). Based on Figures 5.20 and 5.21 of (Bishop 2006b). Figure generated by mixexpDemoOneToMany.

11.2.4 Mixtures of experts

Section 14.7.3 described how to use mixture models in the context of generative classifiers. We can also use them to create discriminative models for classification and regression. For example, consider the data in Figure 11.6(a). It seems like a good model would be three different linear regression functions, each applying to a different part of the input space. We can model this by allowing the mixing weights and the mixture densities to be input-dependent:

$$p(y_i|x_i, z_i = k, \theta) = N(y_i|w_k^T x_i, \sigma^2_k) \quad (11.8)$$

$$p(z_i|x_i, \theta) = \text{Cat}(z_i|S(VT x_i)) \quad (11.9)$$

See Figure 11.7(a) for the DGM. This model is called a mixture of experts or MoE (Jordan and Jacobs 1994). The idea is that each submodel is considered to be an “expert” in a certain region of input space. The function $p(z_i = k|x_i, \theta)$ is called a gating function, and decides which expert to

use, depending on the input values. For example, Figure 11.6(b) shows how the three experts have “carved up” the 1d input space, Figure 11.6(a) shows the predictions of each expert individually (in this case, the experts are just linear regression models), and Figure 11.6(c) shows the overall prediction of the model, obtained using

$$p(y_i|x_i, \theta) = k p(z_i = k|x_i, \theta)p(y_i|z_i = k, \theta) \quad (11.10)$$

We discuss how to fit this model in Section 11.4.3.

It should be clear that we can “plug in” any model for the expert. For example, we can use neural networks (Chapter 16) to represent both the gating functions and the experts. The result is known as a mixture density network. Such models are slower to train, but can be more flexible than mixtures of experts. See (Bishop 1994) for details.

It is also possible to make each expert be itself a mixture of experts. This gives rise to a model known as the hierarchical mixture of experts. See Figure 11.7(b) for the DGM, and Section 16.2.6 for further details.

11.2.4.1 Application to inverse problems

Mixtures of experts are useful in solving inverse problems. These are problems where we have to invert a many-to-one mapping. A typical example is in robotics, where the location of the end effector (hand) y is uniquely determined by the joint angles of the motors, x . However, for any given location y , there are many settings of the joints x that can produce it. Thus the inverse mapping $x = f^{-1}(y)$ is not unique. Another example is kinematic tracking of people from video (Bo et al. 2008), where the mapping from image appearance to pose is not unique, due to self occlusion, etc.

A simpler example, for illustration purposes, is shown in Figure 11.8(a). We see that this defines a function, $y = f(x)$, since for every value x along the horizontal axis, there is a unique response y . This is sometimes called the forwards model. Now consider the problem of computing $x = f^{-1}(y)$. The corresponding inverse model is shown in Figure 11.8(b); this is obtained by simply interchanging the x and y axes. Now we see that for some values along the horizontal axis, there are multiple possible outputs, so the inverse is not uniquely defined. For example, if $y = 0.8$, then x could be 0.2 or 0.8. Consequently, the predictive distribution, $p(x|y, \theta)$ is multimodal.

We can fit a mixture of linear experts to this data. Figure

11.8(b) shows the prediction of each expert, and Figure 11.8(c) shows (a plugin approximation to) the posterior predictive mode and mean. Note that the posterior mean does not yield good predictions. In fact, any model which is trained to minimize mean squared error — even if the model is a flexible nonlinear model, such as neural network — will work poorly on inverse problems such as this. However, the posterior mode, where the mode is input dependent, provides a reasonable approximation.

已思考 9s

下面先把原文完整翻译（不遗漏），再用通俗方式解释，最后给出要点小结。

翻译

图 11.6 (a) 用三条独立的回归直线去拟合同一份数据。(b) 三个不同“专家”的门控函数 (gating functions)。(c) 三个专家预测的按条件加权平均。图由 `mixexpDemo` 生成。

图 11.7 (a) 一个“专家混合” (mixture of experts, MoE)。(b) 分层的专家混合 (hierarchical mixture of experts)。

图 11.8 (a) 来自一个简单“正向模型”的一些数据。(b) 来自其“反问题”模型的一些数据，并用 3 个线性回归的混合进行拟合。训练样本按其责任度着色编码。(c) 预测的均值 (红色叉号) 与众数 (黑色方块)。基于 Bishop(2006b) 的图 5.20 与 5.21。图由 `mixexpDemoOneToMany` 生成。

11.2.4 专家混合 (Mixtures of experts)

14.7.3 节介绍了如何在生成式分类器中使用混合模型。我们也可以用它们来构造用于分类与回归的判别式模型。举例，见图 11.6(a) 的数据：一个理想的模型似乎应是三条不同的线性回归函数，且各自在输入空间的不同区域生效。我们可以通过让混合权重与分量密度依赖输入来建模：

$$p(y_i \mid x_i, z_i = k, \theta) = N(y_i \mid w_k^\top x_i, \sigma_k^2) \quad (11.8)$$

$$p(z_i \mid x_i, \theta) = \text{Cat}(z_i \mid S(V^\top x_i)) \quad (11.9)$$

其 DGM 见图 11.7(a)。该模型称为**专家混合** (MoE; Jordan 与 Jacobs, 1994)。思想是：每个子模型在输入空间的某个区域是一个“专家”。函数 $p(z_i = k \mid x_i, \theta)$ 称为**门控函数**

(gating function) , 根据输入值决定使用哪个专家。例如, 图 11.6(b) 显示三个专家如何把一维输入空间“划分”开来; 图 11.6(a) 显示每个专家单独的预测 (此处专家就是线性回归) ; 图 11.6(c) 显示模型的总体预测, 由下式得到:

$$p(y_i | x_i, \theta) = \sum_k p(z_i = k | x_i, \theta) p(y_i | x_i, z_i = k, \theta) \quad (11.10)$$

我们在 11.4.3 节讨论如何拟合该模型。

显然, 我们可以“插入”任意模型充当专家。例如, 可用神经网络 (第 16 章) 同时表示门控函数与专家, 得到所谓**混合密度网络** (mixture density network) 。这类模型训练更慢, 但比标准 MoE 更灵活; 细节见 Bishop(1994)。

还可以让每个专家自身再是一个专家混合, 从而得到**分层专家混合**; 其 DGM 见图 11.7(b), 更多内容见 16.2.6 节。

11.2.4.1 在反问题中的应用

专家混合在求解**反问题**时很有用。反问题指我们需要对一个**多对一**映射求逆。典型例子出现在机器人中: 末端执行器 (手) 的位姿 y 由电机关节角 x 唯一决定; 然而给定同一个 y , 却可能有多组 x 产生它, 因此逆映射 $x = f^{-1}(y)$ 不是唯一的。另一个例子是视频中的人体运动学跟踪 (Bo 等, 2008) : 由于自遮挡等原因, 从图像外观到姿态的映射也不是唯一的。

更简单的演示例子见图 11.8(a)。这定义了一个函数 $y = f(x)$: 对每个横轴上的 x 都有唯一的 y ——称为**正向模型**。现在考虑计算 $x = f^{-1}(y)$ 。把横纵轴对换得到图 11.8(b) 的**逆模型**; 可以看到, 对某些横轴值会有多个输出, 所以逆映射不唯一。比如 $y = 0.8$ 时, x 可能是 0.2 或 0.8。因此预测分布 $p(x | y, \theta)$ 是**多峰的** (multimodal) 。

我们可以用**线性专家的混合**来拟合这些数据。图 11.8(b) 展示了各专家的预测, 图 11.8(c) 展示了 (一个 plug-in 近似的) 后验预测**众数与均值**。注意: 后验**均值**并不提供好的预测。事实上, 任何以最小化均方误差为训练目标的模型——即使它是非常灵活的非线性模型 (如神经网络) ——在这种反问题上都会表现不佳。相反, **众数** (并且其位置随输入而变) 能够给出更合理的近似。

通俗解释

1) MoE 是什么?

把整个输入空间看成一张地图: 不同区域由**不同专家负责** (例如“左边用直线 1、中间用直线 2、右边用直线 3”)。

- **专家**: 能在某片区域里做出预测的子模型 (这里用线性回归举例)。
- **门控函数**: 像“调度员/路由器”，读入 x 后给出每个专家该被信任的权重 (概率)。这些权重对所有专家求和为 1。
- **总体输出**: 把各专家的预测按门控权重做加权平均 (式 11.10)。这相当于“谁更擅长当前位置，就更听谁的”。

2) 为什么要让权重和分量都依赖输入?

普通混合模型里， π_k 与 p_k 不随 x 变；MoE 里它们随 x 改变，于是可以“按区域切换函数”，从而拟合**分段的、复杂的关系**。

3) 门控函数用什么表示？

常用softmax (**多项 logistic**)：对每个 k ，用 $V^\top x$ 线性打分，再过 softmax，得到 $p(z = k | x)$ 。也可以用神经网络——就成了**混合密度网络**，更灵活但更慢。

4) 分层 MoE

如果一个专家处理的区域仍很复杂，就让它再下分给几个子专家；层层往下，就是**分层专家混合**。它构成了“软分区 + 递归专家”的树状结构。

5) MoE 为什么适合反问题？

反问题的“一个 y 对应多个 x ”导致 $p(x | y)$ **多峰**：平均值落在几个解之间的“中间地带”，往往毫无意义（比如两条可能路径的中线没有真实轨迹）。

- **传统均方误差**会学到**平均值**，所以不行。
- MoE 可以为同一 y 给出多个可能的模式（不同专家负责不同解），于是**众数**（选择最可能的那个）更合理。

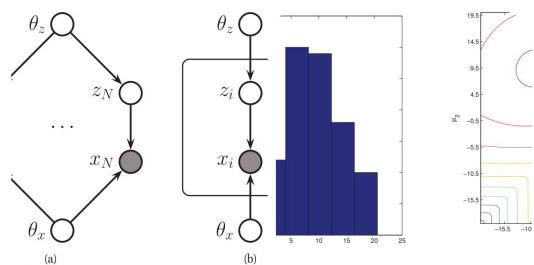
6) 图示快速读法

- 图 11.6：左图三条直线是三个专家；中图三条门控曲线说明各自负责的 x 区域；右图是按门控权重加权后的总预测（可以“平滑”地从一个专家过渡到另一个）。
- 图 11.7：MoE 的结构图；右边是把“专家里再套专家”的分层版本。
- 图 11.8：正向是一一映射；把轴对调得到反问题，多解出现 → 用 MoE 拟合；红叉是均值（不好），黑方块是众数（更符合真实解）。

要点小结

- **MoE 定义**：门控函数 $p(z = k | x)$ 依据输入选择专家；每个专家给出条件分布 $p(y | x, z = k)$ ；总体 $p(y | x) = \sum_k p(z = k | x)p(y | x, z = k)$ 。
- **优势**：对输入空间做**软分区**，能拟合分段/多样的关系；专家与门控都可用 NN 表示（混合密度网络）；还能做**分层**以提升表达力。

- **训练与解释：**门控决定“谁说话更算数”；每个样本对专家有“责任度”；整体预测是各专家按权重的平均。
- **反问题应用：**当 $p(x | y)$ 多峰时，最小化 MSE 的“均值预测”不可用；MoE 能给出多种可能（多位众数），**选择众数**更合理。
- **图像直观：**门控曲线把一维 x 区域切分；不同颜色/直线表示专家的“地盘”；黑色方块（众数）比红色叉（均值）更贴近实际解。



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。

Figure 11.9 A LVM represented as a DGM. Left: Model is unrolled for N examples. Right: same model using plate notation.

Figure 11.10 Left: $N = 200$ data points sampled from a mixture of 2 Gaussians in 1d, with $\pi_k = 0.5$, $\sigma_k = 5$, $\mu_1 = -10$ and $\mu_2 = 10$. Right: Likelihood surface $p(D|\mu_1, \mu_2)$, with all other parameters set to their true values. We see the two symmetric modes, reflecting the unidentifiability of the parameters. Figure generated by mixGaussLikSurfaceDemo.

11.3 Parameter estimation for mixture models

We have seen how to compute the posterior over the hidden variables given the observed variables, assuming the parameters are known. In this section, we discuss how to learn the parameters. In Section 10.4.2, we showed that when we have complete data and a factored prior, the posterior over the parameters also factorizes, making computation very simple. Unfortunately this is no longer true if we have hidden variables and/or missing data. The reason is apparent from looking at Figure 11.9. If the z_i were observed, then by d-separation, we see that $\theta_z \perp \theta_x | D$, and hence the posterior will factorize. But since, in

an LVM, the z_i are hidden, the parameters are no longer independent, and the posterior does not factorize, making it much harder to compute. This also complicates the computation of MAP and ML estimates, as we discuss below.

11.3.1 Unidentifiability

The main problem with computing $p(\theta|D)$ for an LVM is that the posterior may have multiple modes. To see why, consider a GMM. If the z_i were all observed, we would have a unimodal posterior for the parameters:

$$p(\theta|D) = \text{Dir}(\pi|D) \prod_{k=1}^K \text{NIW}(\mu_k, \Sigma_k|D) \quad (11.11)$$

Consequently we can easily find the globally optimal MAP estimate (and hence globally optimal MLE). But now suppose the z_i 's are hidden. In this case, for each of the possible ways of "filling in" the z_i 's, we get a different unimodal likelihood. Thus when we marginalize out over the z_i 's, we get a multi-modal posterior for $p(\theta|D)$.¹ These modes correspond to different labelings of the clusters.

This is illustrated in Figure 11.10(b), where we plot the likelihood function, $p(D|\mu_1, \mu_2)$, for a 2D GMM with $K = 2$ for the data is shown in Figure 11.10(a). We see two peaks, one corresponding to the case where $\mu_1 = -10, \mu_2 = 10$, and the other to the case where $\mu_1 = 10, \mu_2 = -10$. We say the parameters are not identifiable, since there is not a unique MLE. Therefore there cannot be a unique MAP estimate (assuming the prior does not rule out certain labelings), and hence the posterior must be multimodal.

The question of how many modes there are in the parameter posterior is hard to answer. There are $K!$ possible labelings, but some of the peaks might get merged. Nevertheless, there can be an exponential number, since finding the optimal MLE for a GMM is NP-hard (Aloise et al. 2009; Drineas et al. 2004).

Unidentifiability can cause a problem for Bayesian inference. For example, suppose we draw some samples from the posterior, $\theta(s) \sim p(\theta|D)$, and then average them, to try to approximate the posterior mean, $\theta = \frac{1}{S} \sum_{s=1}^S \theta(s)$. (This kind of Monte Carlo approach is explained in more detail in Chapter 24.) If the samples come from different modes, the average will be meaningless. Note, however, that it is reasonable to average the posterior predictive distributions, $p(x) \approx \frac{1}{S} \sum_{s=1}^S p(x|\theta(s))$, since the likelihood function is invariant to which mode the

parameters came from. A variety of solutions have been proposed to the unidentifiability problem. These solutions depend on the details of the model and the inference algorithm that is used. For example, see (Stephens 2000) for an approach to handling unidentifiability in mixture models using MCMC.

The approach we will adopt in this chapter is much simpler: we just compute a single local mode, i.e., we perform approximate MAP estimation. (We say “approximate” since finding the globally optimal MLE, and hence MAP estimate, is NP-hard, at least for mixture models (Aloise et al. 2009).) This is by far the most common approach, because of its simplicity. It is also a reasonable approximation, at least if the sample size is sufficiently large. To see why, consider Figure 11.9(a). We see that there are N latent variables, each of which gets to “see” one data point each. However, there are only two latent parameters, each of which gets to see N data points. So the posterior uncertainty about the parameters is typically much less than the posterior uncertainty about the latent variables. This justifies the common strategy of computing $p(z_i|x_i, \hat{\theta})$, but not bothering to compute $p(\theta|D)$. In Section 5.6, we will study hierarchical Bayesian models, which essentially put structure on top of the parameters. In such models, it is important to model $p(\theta|D)$, so that the parameters can send information between themselves. If we used a point estimate, this would not be possible

已思考 6s

下面先把原文完整翻译（不遗漏），再用通俗解释，最后给出要点小结。

翻译

图 11.9 把一个潜变量模型 (LVM) 表示成有向图模型 (DGM)。左：把模型对 N 个样本“展开”。右：同一模型的“板式 (plate) ”记法。

图 11.10 左：从一维、2 个高斯的混合中抽取的 $N = 200$ 个数据点，取 $\pi_k = 0.5$, $\sigma_k = 5$, $\mu_1 = -10$, $\mu_2 = 10$ 。右：似然曲面 $p(D | \mu_1, \mu_2)$ (其余参数都设为真实值)。可以看

到两个对称的极大值（模态），反映了参数的**不可辨识性**。图由 `mixGaussLikSurfaceDemo` 生成。

11.3 混合模型的参数估计

前面我们已经在**已知参数**的条件下，讨论了“在给定观测变量时，如何计算隐藏变量的后验分布”。本节讨论**如何学习参数**。在 10.4.2 节中我们展示过：当我们拥有**完整数据**且使用**可分解的先验**时，参数的后验也会**因子化**，于是计算很简单。不幸的是，一旦存在**隐藏变量**和/或**缺失数据**，这一点就不再成立。原因从图 11.9 很容易看出：如果 z_i 都被观测到了，那么根据 d-分离 (d-separation)，有

$$\theta_z \perp \theta_x \mid D,$$

因此后验会因子化。但在 LVM 中， z_i 是隐藏的，于是参数之间不再独立，后验也不再因子化，计算就困难得多。下面要讨论的 MAP 和 ML 估计也会因此变得更复杂。

11.3.1 不可辨识性 (Unidentifiability)

对 LVM 计算 $p(\theta \mid D)$ 的主要问题在于：**后验可能具有多个模态**。为什么？来看 GMM。如果所有 z_i 都已观测到，则参数的后验是单峰（单模态）的：

$$p(\theta \mid D) = \text{Dir}(\pi \mid D) \prod_{k=1}^K \text{NIW}(\boldsymbol{\mu}_k, \Sigma_k \mid D) \quad (11.11)$$

因此我们很容易找到**全局最优的 MAP**（也就有**全局最优的 MLE**）。但现在假设 z_i 是隐藏的。此时，对每一种可能的“填充 z_i ”的方式，我们都会得到一个不同的**单峰似然**。于是对 z_i 积分（边缘化）之后， $p(\theta \mid D)$ 就会变成**多峰的后验**。¹ 这些峰对应着对簇作不同“标号”的情形。

这在图 11.10(b) 中得到了说明：我们为图 11.10(a) 的数据绘制了 $K = 2$ 的 GMM 的似然函数 $p(D \mid \mu_1, \mu_2)$ 。可以看到两个峰，一个对应 $\mu_1 = -10, \mu_2 = 10$ ，另一个对应 $\mu_1 = 10, \mu_2 = -10$ 。我们说这些参数**不可辨识**，因为**极大似然解不唯一**。因此，除非先验排除了某些标号方式，否则 MAP 也不唯一，后验必然是多峰的。参数后验里到底有多少个模态并不容易回答。理论上有 $K!$ 种标号方式，但有些峰可能会合并。尽管如此，模态个数可能呈指数级，因为 GMM 寻找最优 MLE 是 NP-hard 的 (Aloise 等, 2009; Drineas 等, 2004)。

不可辨识性会给贝叶斯推断带来麻烦。比如，我们从后验中采样 $\theta^{(s)} \sim p(\theta \mid D)$ ，然后把它们平均来近似后验均值

$$\bar{\theta} = \frac{1}{S} \sum_{s=1}^S \theta^{(s)}.$$

（这种蒙特卡洛方法在第 24 章会详细解释。）如果样本来自不同的模态，这个**参数平均**就没有意义。不过，**平均后验预测分布**是合理的，即

$$p(x) \approx \frac{1}{S} \sum_{s=1}^S p(x | \theta^{(s)}),$$

因为似然对“参数来自哪个模态”是不变的。

针对不可辨识性，文献里提出了多种解决办法，依赖具体模型与推断算法。比如 Stephens(2000) 给出了在混合模型中用 MCMC 处理不可辨识性的方式。

本章采取更简单的思路：只求一个局部模态，也就是做**近似 MAP 估计**。（之所以说“近似”，是因为寻找全局最优的 MLE、进而 MAP，在混合模型里本身就是 NP-hard 的

(Aloise 等, 2009)。）由于简单，这也是最常见的做法；当样本量足够大时，它通常也是合理的近似。原因如下（见图 11.9(a)）：有 N 个潜变量，它们各自只“看到”一个数据点；而只有两个潜在参数，它们“看到”的是所有 N 个数据点。因此，**参数后验的不确定性通常远小于潜变量的不确定性**。这就为一种常见策略提供了依据：只计算 $p(z_i | x_i, \hat{\theta})$ ，而不去计算 $p(\theta | D)$ 。

在 5.6 节我们会研究**分层贝叶斯模型**，它本质上在参数上再加了一层结构。在那样的模型里，刻画 $p(\theta | D)$ 就很重要，因为参数之间需要**互相传递信息**；如果只用一个点估计，这是做不到的。

¹ 原文这里的“1”是脚注标记；脚注内容不在给定片段中。

通俗解释

1) 为什么有隐藏变量时，参数学习更难？

- 如果 z_i 都是“已知标签”，那么“管 z 的参数”和“管 x 的参数”彼此独立，后验能分开算，像把两件作业分给两个人各做各的。
- 但 z_i 看不见时，两类参数都要“解释”同一批数据，于是**缠在一起**，后验不能分开，计算就麻烦了。

2) 什么是“不可辨识”？

- 混合模型里，“簇 1 / 簇 2”只是**名字**，调换名字不会改变整体分布。
- 因而似然（或后验）会出现多个**对称的高峰**：把 μ_1, μ_2 互换一样好。
- 这意味着“最优参数”不唯一 → MLE/MAP 都可能不唯一 → 后验多峰。

3) 为什么不能简单地“平均参数”？

- 如果从不同峰抽到的参数直接平均，结果往往处在两个好解中间，**毫无意义**。
- 但**平均预测**是可以的：无论参数来自哪个峰，算出来的 $p(x | \theta)$ 加权平均仍是合理的预测。

4) 现实中我们怎么做？

- 通常只求一个局部最优（近似 MAP）。理由：
 - 每个参数是用很多数据估出来的，**不确定性小**；
 - 每个 z_i 只看见一个样本，**不确定性大**；
 - 所以只用 $\hat{\theta}$ 去算 $p(z_i | x_i, \hat{\theta})$ 往往已经够用了。
- 但在**分层贝叶斯里**，参数之间需要“交流”（通过它们的后验来传递信息），这时就必须对 $p(\theta | D)$ 建模，点估计不够。

5) 图 11.10 的直观

- 左图数据明显来自两个峰（一个在 -10，一个在 10）。
- 右图“似然地形”有两个等高的山峰： $(\mu_1, \mu_2) = (-10, 10)$ 与 $(10, -10)$ → 说明**标签互换导致多峰**。

6) 板式记法（图 11.9 右）

- 大方框（plate）里画的是会对 $i = 1, \dots, N$ **重复**的结构：每个样本有自己的 z_i, x_i 。
- 方框外的 θ_z, θ_x 是**共享参数**，连接到所有样本。

要点小结

- 有隐藏变量时，参数的后验 **不再因子化**，计算更难。
- 混合模型的参数通常**不可辨识**：对称标号（label switching）导致**多峰后验/似然**；寻找全局 MLE 是 NP-hard。
- 不能把来自不同模态的**参数**直接平均；但可以平均**预测分布**。
- 实务中常做**近似 MAP（单个局部模态）**，再用 $\hat{\theta}$ 来推断 $p(z_i | x_i, \hat{\theta})$ 。
- 当使用**分层贝叶斯**时，需要刻画 $p(\theta | D)$ （让参数之间共享/传递信息）；此时点估计不够用。
- 板式记法简洁表达“对 N 个样本重复的结构”，参数在板外、对所有样本共享。

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。

11.3.2 Computing a MAP estimate is non-convex
 In the previous sections, we have argued, rather heuristically, that the likelihood function has multiple modes, and hence that finding an MAP or ML estimate will be hard. In this section, we show this result by more

algebraic means, which sheds some additional insight into the problem. Our presentation is based in part on (Rennie 2004). Consider the log-likelihood for an LVM:

$$\log p(D|\theta) = \sum_i \log p(z_i | x_i, \theta) \quad (11.12)$$

Unfortunately, this objective is hard to maximize, since we cannot push the log inside the sum. This precludes certain algebraic simplifications, but does not prove the problem is hard. Now suppose the joint probability distribution $p(z_i, x_i | \theta)$ is in the exponential family, which means it can be written as follows:

$$p(x, z | \theta) = \frac{1}{Z(\theta)} \exp[\theta^T \varphi(x, z)] \quad (11.13)$$

where $\varphi(x, z)$ are the sufficient statistics, and $Z(\theta)$ is the normalization constant (see Section 9.2 for more details).

It can be shown (Exercise 9.2) that the MVN is in the exponential family, as are nearly all of the distributions we have encountered so far, including Dirichlet, multinomial, Gamma, Wishart, etc. (The Student distribution is a notable exception.) Furthermore, mixtures of exponential families are also in the exponential family, providing the mixing indicator variables are observed (Exercise 11.11).

With this assumption, the complete data log likelihood can be written as follows:

$$c(\theta) = \sum_i \log p(x_i, z_i | \theta) = \theta^T (\sum_i \varphi(x_i, z_i)) - N Z(\theta) \quad (11.14)$$

The first term is clearly linear in θ . One can show that $Z(\theta)$ is a convex function (Boyd and Vandenberghe 2004), so the overall objective is concave (due to the minus sign), and hence has a unique maximum. Now consider what happens when we have missing data. The observed data log likelihood is given by

$$c(\theta) = \sum_i \log p(x_i, z_i | \theta) = \sum_i \log p(x_i | z_i, \theta) + \sum_i \log p(z_i | x_i, \theta) - N \log Z(\theta) \quad (11.15)$$

One can show that the log-sum-exp function is convex (Boyd and Vandenberghe 2004), and we know that $Z(\theta)$ is convex. However, the difference of two convex functions is not, in general, convex. So the objective is neither convex nor concave, and has local optima. The disadvantage of non-convex functions is that it is usually hard to find their global optimum. Most optimization algorithms will only find a local optimum; which one they find depends on where they start. There are some algorithms, such as simulated annealing (Section 24.6.1) or genetic algorithms, that claim to always find the global optimum, but this is only under unrealistic assumptions.

(e.g., if they are allowed to be cooled “infinitely slowly”, or allowed to run “infinitely long”). In practice, we will run a local optimizer, perhaps using multiple random restarts to increase our chance of finding a “good” local optimum. Of course, careful initialization can help a lot, too. We give examples of how to do this on a case-by-case basis. Note that a convex method for fitting mixtures of Gaussians has been proposed. The idea is to assign one cluster per data point, and select from amongst them, using a convex 1-type penalty, rather than trying to optimize the locations of the cluster centers. See (Lashkari and Golland 2007) for details. This is essentially an unsupervised version of the approach used in sparse kernel logistic regression, which we will discuss in Section 14.3.2. Note, however, that the 1 penalty, although convex, is not necessarily a good way to promote sparsity, as discussed in Chapter 13. In fact, as we will see in that Chapter, some of the best sparsity-promoting methods use non-convex penalties, and use EM to optimize them! The moral of the story is: do not be afraid of non-convexity.

已思考 13s

下面先把原文完整翻译（不遗漏），再用通俗解释，最后给出要点小结。

翻译

11.3.2 计算 MAP 估计是一个非凸问题

在前面的章节里，我们用较为直观的方式说明了：似然函数有多个模态，因此要找到 MAP 或 ML 估计会很困难。本节用更代数化的方法来证明这一点，以便对问题获得更多洞见。我们的表述部分基于 (Rennie 2004)。

考虑某个潜变量模型 (LVM) 的对数似然：

$$\log p(\mathbf{D} \mid \theta) = \sum_i \log \sum_{z_i} p(x_i, z_i \mid \theta) \quad (11.12)$$

遗憾的是，这个目标函数很难最大化，因为我们不能把对数“推进到求和里面”。这使得某些代数化简无法进行，但这本身还不能证明问题就是困难的。

现在假设联合分布 $p(z, x \mid \theta)$ 属于指数族，也就是说它可以写成

$$p(x, z | \theta) = \frac{1}{Z(\theta)} \exp[\theta^T \phi(x, z)] \quad (11.13)$$

其中 $\phi(x, z)$ 是充分统计量， $Z(\theta)$ 是归一化常数（更多细节见 9.2 节）。可以证明（练习 9.2）多元高斯属于指数族；我们至今遇到的几乎所有分布也都属于指数族，包括 Dirichlet、多项式（multinomial）、Gamma、Wishart 等等（Student-t 分布是一个显著的例外）。此外，只要混合指示变量是已观测的，指数族的混合仍然属于指数族（练习 11.11）。在这个假设下，完备数据的对数似然可以写成

$$c(\theta) = \sum_i \log p(x_i, z_i | \theta) = \theta^T (\sum_i \phi(x_i, z_i)) - N Z(\theta) \quad (11.14)$$

第一项显然对 θ 是线性的。可以证明 $Z(\theta)$ 是一个**凸函数**（Boyd and Vandenberghe, 2004），因此整体目标是**凸的**（因为有负号），从而有**唯一的最大值**。

现在看看有缺失数据时会发生什么。**观测数据**的对数似然为

$$\ell(\theta) = \sum_i \log \sum_{z_i} p(x_i, z_i | \theta) = \sum_i \log \sum_{z_i} e^{\theta^T \phi(z_i, x_i)} - N \log Z(\theta) \quad (11.15)$$

可以证明，**log-sum-exp** 函数是**凸的**（Boyd and Vandenberghe, 2004），并且我们知道 $\log Z(\theta)$ 也是凸的。然而，**两个凸函数之差一般既不凸也不凹**。因此该目标既非凸也非凹，会存在**局部最优**。

非凸函数的缺点在于通常很难找到**全局最优**。大多数优化算法只能找到某个**局部最优**；它找到哪个，取决于初始点。确实有一些算法（比如模拟退火，见 24.6.1 节，或遗传算法）宣称总能找到全局最优，但那只在一些不现实的假设下成立（例如允许“无限慢”降温，或允许“无限长”运行）。在实践中，我们通常运行一个**局部优化器**，并结合**多次随机重启**来提高找到“较好”局部最优的机会。当然，**精心初始化**也很有帮助。我们会按具体问题给出初始化的例子。

需要指出的是，确实有人提出了一个用于拟合高斯混合的**凸方法**。其思想是：给每个数据点都分配一个簇，然后在这些簇中进行选择，使用一种** ℓ_1 型惩罚，而不是直接优化簇中心的位置。细节见（Lashkari and Golland, 2007）。这本质上是第 14.3.2 节要讨论的“稀疏核逻辑回归”方法的无监督版本**。不过要注意，虽然 ℓ_1 惩罚是凸的，但它未必是促进稀疏性的最佳方式（见第 13 章的讨论）。事实上，正如我们将在该章看到的，某些最有效的“促稀疏”方法使用的是**非凸惩罚**，并且用 EM 来优化它们！这个故事的寓意是：**不要害怕非凸性**。

通俗解释

1) 为什么“有隐藏变量就变难”?

- 完备数据时 (z_i 都已知)，指数族模型的对数似然 = “线性项 - 凸函数”。线性减凸 \Rightarrow 凹函数，只有一个最高点 \Rightarrow 很好解。
- 缺失 z_i 时，需要先把不同 z_i 的可能性相加再取对数 (log-sum-exp)。这部分是凸的；再减去另一项凸函数。凸 - 凸 通常既不凸也不凹 \Rightarrow 会出现多个“山峰”，也就是多个局部极值。

2) log-sum-exp 是谁？

$\text{lse}(a_1, \dots, a_K) = \log \sum_k e^{a_k}$ 。它是“softmax 的母函数”，平滑地近似 $\max_k a_k$ ，并且是凸函数。

3) 非凸带来的实际后果

- 典型算法（如 EM、梯度法）会依赖初值，可能停在不同的局部最优。
- 你可以做：
 - 多次随机重启；
 - 用有经验的初始化（如 K-means/++ 先找簇中心，再进入 EM）；
 - 适度正则化、防止退化；
 - 监控多个运行、选对数似然较高的结果。

4) “有没有办法把它弄成凸的？”

- 有研究用“每点一簇 + ℓ_1 选择”的方式，把问题改写成凸优化；但这改变了原问题的结构，也未必得到最合适密度模型。
- 而且，尽管 ℓ_1 是凸的，它并不总是最好的稀疏先验；不少更强的稀疏方法本身就是非凸的（常配合 EM 进行优化）。

5) 结论

- 在混合模型/潜变量模型里，MAP/ML 优化本质上是非凸的。
- 这不是坏事：通过良好初始化 + 多次重启 + 合理正则化，往往就能得到足够好的解；理论和实践中也有很多“拥抱非凸”的成功范例。

要点小结

- 完备数据 + 指数族：对数似然 $c(\theta) = \theta^\top \phi - NZ(\theta)$ 为凹 \Rightarrow 唯一最大值。
- 隐藏数据： $\ell(\theta) = \sum \log \sum_{z_i} e^{\theta^\top \phi} - N \log Z(\theta)$ 是凸 - 凸 \Rightarrow 非凸 \Rightarrow 多个局部最优。
- 非凸 \Rightarrow 依赖初值；实际做法：局部优化 + 多次重启 + 好的初始化（如 K-means/++）+ 正则化。

- 也有“凸化”的替代思路（每点一族 + ℓ_1 选择），但 ℓ_1 并非总是最佳稀疏方法；很多强力稀疏方法本身是**非凸**并用 EM 优化。
- **核心信息：**在 LVM/混合模型里，MAP/ML 优化**非凸**是常态——**不要恐惧非凸**，而要用合适的工程策略去对付它。

Model	Section
Mix. Gaussians	11.4.2
Mix. experts	11.4.3
Factor analysis	12.1.5
Student T	11.4.5
Probit regression	11.4.6
DGM with hidden variables	11.4.4
MVN with missing data	11.6.1
HMMs	17.5.2
Shrinkage estimates of Gaussian means	Exercise 11.13

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。

Table 11.2 Some models discussed in this book for which EM can be easily applied to find the ML/ MAP parameter estimate.

11.4 The EM algorithm

For many models in machine learning and statistics, computing the ML or MAP parameter estimate is easy provided we observe all the values of all the relevant random variables, i.e., if we have complete data. However, if we have missing data and/or latent variables, then computing the ML/MAP estimate becomes hard. One approach is to use a generic gradient-based optimizer to find a local minimum of the negative log likelihood or NLL, given by

$$\text{NLL}(\theta) = -\frac{1}{N} \sum_{i=1}^N \log p(D_i | \theta) \quad (11.16)$$

However, we often have to enforce constraints, such as the fact that covariance matrices must be positive definite, mixing weights must sum to one, etc., which can be tricky (see Exercise 11.5). In such cases, it is often much simpler (but not always faster) to use an algorithm called expectation maximization, or EM for short (Dempster et al. 1977; Meng and van Dyk 1997; McLachlan and Krishnan 1997). This is a simple iterative algorithm, often with

closed-form updates at each step. Furthermore, the algorithm automatically enforce the required constraints. EM exploits the fact that if the data were fully observed, then the ML/ MAP estimate would be easy to compute. In particular, EM is an iterative algorithm which alternates between inferring the missing values given the parameters (E step), and then optimizing the parameters given the “filled in” data (M step). We give the details below, followed by several examples. We end with a more theoretical discussion, where we put the algorithm in a larger context. See Table 11.2 for a summary of the applications of EM in this book.

11.4.1 Basic idea

Let x_i be the visible or observed variables in case i , and let z_i be the hidden or missing variables. The goal is to maximize the log likelihood of the observed data:

$$(\theta) = \sum_{i=1}^N \log p(x_i|\theta) = \sum_{i=1}^N \log z_i p(x_i, z_i|\theta) \quad (11.17)$$

Unfortunately this is hard to optimize, since the log cannot be pushed inside the sum.

EM gets around this problem as follows. Define the complete data log likelihood to be

$$c(\theta) = \sum_{i=1}^N \log p(x_i, z_i|\theta) \quad (11.18)$$

This cannot be computed, since z_i is unknown. So let us define the expected complete data log likelihood as follows:

$$Q(\theta, \theta_{t-1}) = \mathbb{E} c(\theta) | D, \theta_{t-1} \quad (11.19)$$

where t is the current iteration number. Q is called the auxiliary function. The expectation is taken wrt the old parameters, θ_{t-1} , and the observed data D . The goal of the Estep is to compute $Q(\theta, \theta_{t-1})$, or rather, the terms inside of it which the MLE depends on; these are known as the expected sufficient statistics or ESS. In the Mstep, we optimize the Q function wrt θ :

$$\theta_t = \operatorname{argmax}_{\theta} Q(\theta, \theta_{t-1}) \quad (11.20)$$

To perform MAP estimation, we modify the M step as follows:

$$\theta_t = \operatorname{argmax}_{\theta} Q(\theta, \theta_{t-1}) + \log p(\theta) \quad (11.21)$$

The E step remains unchanged. In Section 11.4.7 we show that the EM algorithm monotonically increases the log likelihood of the observed data (plus the log prior, if doing MAP estimation), or it stays the same. So if the

objective ever goes down, there must be a bug in our math or our code. (This is a surprisingly useful debugging tool!) Below we explain how to perform the E and M steps for several simple models, that should make things clearer.

下面先把原文逐字不遗漏地译成中文，然后用通俗方式解释，最后给出要点小结。

翻译

表 11.2 本书中一些可以较容易地用 EM 寻找 ML/MAP 参数估计的模型与对应小节。

- 混合高斯 (Mix. Gaussians) ——11.4.2
 - 专家混合 (Mix. experts) ——11.4.3
 - 因子分析 (Factor analysis) ——12.1.5
 - 学生-t (Student T) ——11.4.5
 - Probit 回归 (Probit regression) ——11.4.6
 - 含隐藏变量的有向图模型 (DGM with hidden variables) ——11.4.4
 - 有缺失数据的多元高斯 (MVN with missing data) ——11.6.1
 - 隐马尔可夫模型 (HMMs) ——17.5.2
 - 高斯均值的收缩估计 (Shrinkage estimates of Gaussian means) ——练习 11.13
-

11.4 EM 算法

对许多机器学习与统计模型来说，只要我们观测到了所有相关随机变量的取值（即**完整数据**），计算 ML 或 MAP 的参数估计就很容易。然而，一旦存在**缺失数据**和/或**潜变量**，计算 ML/MAP 就会变难。一个思路是用通用的基于梯度的优化器，去最小化**负对数似然** (negative log-likelihood, NLL)：

$$\text{NLL}(\theta) = -\frac{1}{N} \log p(\mathbf{D} \mid \theta) \quad (11.16)$$

但我们常常还得满足一些约束，例如协方差矩阵必须正定、混合权重要和为 1 等，这会比较棘手（见练习 11.5）。在这样的情形下，常常更简单（但不一定更快）的是使用**期望最大化**算法，简称 EM (Dempster 等 1977; Meng & van Dyk 1997; McLachlan & Krishnan 1997)。这是一种简单的迭代算法，很多模型每一步都有**闭式更新**。而且，该算法会自动满足所需的约束。

EM 利用了这样的事实：**如果数据是完全观测的**，那么 ML/MAP 的计算是容易的。具体地说，EM 是一个交替迭代：在已知参数的条件下**推断缺失值**（E 步），然后在“补齐”的数据下**优化参数**（M 步）。我们在下文给出细节与若干示例，并在最后进行更理论的讨论，把该算法放在更大的背景下来理解。本书中 EM 的应用小结见表 11.2。

11.4.1 基本思想

令 x_i 为第 i 个样本的可见/观测变量， z_i 为隐藏/缺失变量。我们的目标是最大化**观测数据的对数似然**：

$$\ell(\theta) = \sum_{i=1}^N \log p(x_i | \theta) = \sum_{i=1}^N \log \sum_{z_i} p(x_i, z_i | \theta) \quad (11.17)$$

遗憾的是，这个目标不易优化，因为对数不能推进到求和符号内部。

EM 的绕法如下。把**完备数据**（即把 z_i 也当作已知）的对数似然定义为

$$c(\theta) = \sum_{i=1}^N \log p(x_i, z_i | \theta) \quad (11.18)$$

但由于 z_i 未知，我们计算它的**期望的完备数据对数似然**：

$$Q(\theta, \theta^{t-1}) = E_{(z|D, \theta^{t-1})}[c(\theta)] \quad (11.19)$$

其中 t 是当前迭代次数。 Q 被称为**辅助函数**。期望是相对于旧参数 θ^{t-1} 与观测数据 D 取的。E 步的目标是计算 $Q(\theta, \theta^{t-1})$ ，或者更确切地说，计算**极大似然所依赖的那些项**——它们叫作**期望充分统计量** (ESS)。在 M 步，我们对 θ 最大化 Q ：

$$\theta^t = \arg \max_{\theta} Q(\theta, \theta^{t-1}) \quad (11.20)$$

若要做 MAP 估计，M 步改为

$$\theta^t = \arg \max_{\theta} \{Q(\theta, \theta^{t-1}) + \log p(\theta)\} \quad (11.21)$$

E 步不变。11.4.7 节将证明：EM 算法会**单调增大**观测数据的对数似然（若做 MAP，则增大对数后验），或至少不下降。因此如果目标值下降了，要么我们的推导有错，要么代码有 bug。（这其实是个非常好用的调试工具！）下面我们会在若干简单模型上展示如何具体执行 E/M 两步，以帮助理解。

通俗解释

1) NLL 是什么? 为什么用 EM?

- NLL 就是“把概率改写成要最小化的损失”：概率越大，NLL 越小。
- 用通用优化器直接最小化 NLL 常会被**约束**搞得很麻烦（例如混合权重必须非负且和为 1、协方差要正定）。
- EM 把“难点（求和里的对数）”变成“先求期望再最大化”，很多模型由此得到**简单的闭式更新**，并且天然满足约束（比如 softmax 得到的权重自动非负且和为 1）。

2) E 步在做什么?

- 先用**旧的参数** θ^{t-1} 来估计隐藏量 z 的后验分布（例如在 GMM 中的每个簇的责任度 r_{ik} ）。
- 再用这些后验的**期望**去“填”到完备数据对数似然里，得到 $Q(\theta, \theta^{t-1})$ 。
- 直观地说：**猜一猜**每个样本属于哪个隐藏状态、或缺失值大概是多少。

3) M 步在做什么?

- 把上一步“猜出的软标签/填充值”当成已知数据，重新**最大化**参数。这一步往往是熟悉的封闭解：
 - 例如 GMM 中，高斯均值是“带权平均”、协方差是“带权协方差”、混合权重是“带权频率”。
- 如做 MAP，就在 M 步里把先验 $\log p(\theta)$ 加进去，相当于正则化。

4) 为什么 EM 会单调提升对数似然?

- 关键是 Q 函数下界了真实的 $\ell(\theta)$ （可由 Jensen 不等式或 ELBO 观点推导）；
- E 步把下界**紧贴**在当前参数处，M 步把下界**抬高**；每次迭代都不降。

5) 和“直接优化 NLL”的对比

- 直接优化：要处理复杂约束、可能没有闭式梯度步、调参繁琐。
- EM：每步更“结构化”，常常算责任度→求带权统计量→闭式更新，工程上更稳、更易调试（看似然是否单调上升即可）。

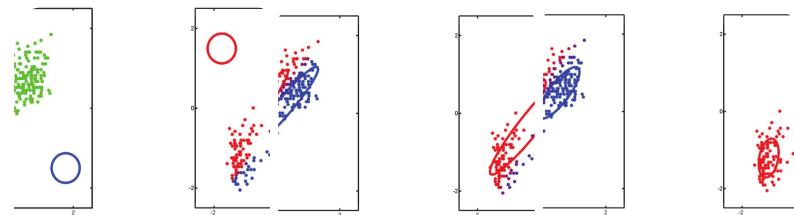
6) 一句话理解公式 (11.19) 的 Q

- $Q(\theta, \theta^{t-1}) =$ “用旧参数把看不见的东西**先补成概率版的软标签**，再按**补好的数据**去评估如果参数变成 θ 时的完备数据对数似然期望”。
- 所以 E 步 = 计算软标签/期望充分统计量；M 步 = 用这些“软统计量”更新参数。

要点小结

- EM 适用场景：有**潜变量/缺失值**导致 $\log \sum$ 结构难以直接最大化的模型（表 11.2 给出常见清单：GMM、MoE、因子分析、HMM、缺失高斯等）。
- 核心循环：

- **E 步:** $Q(\theta, \theta^{t-1}) = E_{z|D, \theta^{t-1}}[c(\theta)]$, 计算**期望充分统计量** (如责任度/软计数/带权和)。
- **M 步:** $\theta^t = \arg \max_{\theta} Q(\theta, \theta^{t-1})$ (MAP 时再加 $\log p(\theta)$)。
- **优势:** 常有**闭式更新**、自动满足约束、**对数似然单调不降** (是个好用的“自带断言”调试器)。
- **直觉:** EM = “先用旧参数猜隐藏值 (软标签)，再当作已知来**更新参数**”，反复交替直到收敛。
- **实践提示:** 初始化重要 (如用 K-means/++ 初始化 GMM 的均值和权重)；每步检查对数似然是否上升。



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。

Figure 11.11 Illustration of the EM for a GMM applied to the Old Faithful data. (a) Initial (random) values of the parameters. (b) Posterior responsibility of each point computed in the first E step. The degree of redness indicates the degree to which the point belongs to the red cluster, and similarly for blue; these purple points have a roughly uniform posterior over clusters. (c) We show the updated parameters after the first M step. (d) After 3 iterations. (e) After 5 iterations. (f) After 16 iterations. Based on (Bishop 2006a) Figure 9.8. Figure generated by mixGaussDemoFaithful.

11.4.2 EM for GMMs

In this section, we discuss how to fit a mixture of Gaussians using EM. Fitting other kinds of mixture models requires a straightforward modification — see Exercise 11.3. We assume the number of mixture components, K , is known (see Section 11.5 for discussion of this point).

11.4.2.1 Auxiliary function

The expected complete data log likelihood is given by

$$Q(\theta, \theta(t-1)) = E_i \log p(x_i, z_i | \theta) \quad (11.22)$$

$$= i E \log K \sum_{k=1}^K (\pi_k p(x_i | \theta_k)) I(z_i = k) \quad (11.23)$$

$$= i \sum_k E[I(z_i = k)] \log [\pi_k p(x_i | \theta_k)] \quad (11.24)$$

$$= i \sum_k p(z_i = k | x_i, \theta(t-1)) \log [\pi_k p(x_i | \theta_k)] \quad (11.25)$$

$$= i \sum_k r_{ik} \log \pi_k + i \sum_k r_{ik} \log p(x_i | \theta_k) \quad (11.26)$$

where $r_{ik} = p(z_i = k | x_i, \theta(t-1))$ is the responsibility that cluster k takes for data point i . This is computed in the E step, described below.

11.4.2.2 E step

The E step has the following simple form, which is the same for any mixture model:

$$r_{ik} = \frac{\pi_k p(x_i | \theta(t-1))}{\sum_{k'} \pi_{k'} p(x_i | \theta(t-1))} \quad (11.27)$$

11.4.2.3 M step

In the M step, we optimize Q wrt π and the θ_k . For π , we obviously have

$$\pi_k = \frac{1}{N} \sum_i r_{ik} = r_k / N \quad (11.28)$$

where $r_k = \sum_i r_{ik}$ is the weighted number of points assigned to cluster k . To derive the M step for the μ_k and Σ_k terms, we look at the parts of Q that depend on μ_k and Σ_k . We see that the result is

$$(\mu_k, \Sigma_k) = \sum_i r_{ik} \log p(x_i | \theta_k) \quad (11.29)$$

$$= -\frac{1}{2} \sum_i r_{ik} \log |\Sigma_k| + (x_i - \mu_k)^T \Sigma_k^{-1} (x_i - \mu_k) \quad (11.30)$$

This is just a weighted version of the standard problem of computing the MLEs of an MVN (see Section 4.1.3). One can show (Exercise 11.2) that the new parameter estimates are given by

$$\mu_k = \frac{1}{r_k} \sum_i r_{ik} x_i \quad (11.31)$$

$$\Sigma_k = \frac{1}{r_k} \sum_i r_{ik} (x_i - \mu_k)(x_i - \mu_k)^T \quad (11.32)$$

These equations make intuitive sense: the mean of cluster k is just the weighted average of all points assigned to cluster k , and the covariance is proportional to the weighted empirical scatter matrix. After computing the new estimates, we set $\theta_t = (\pi_k, \mu_k, \Sigma_k)$ for $k = 1: K$, and go to the next E step. 11.4.2.4 Example

An example of the algorithm in action is shown in Figure 11.11. We start with $\mu_1 = (-1, 1)$, $\Sigma_1 = I$, $\mu_2 = (1, -1)$, $\Sigma_2 = I$. We color code points such that blue points come from cluster 1 and red points from cluster 2. More precisely, we set the color to

$$\text{color}(i) = r_{i1}\text{blue} + r_{i2}\text{red} \quad (11.33)$$

so ambiguous points appear purple. After 20 iterations,

the algorithm has converged on a good clustering. (The data was standardized, by removing the mean and dividing by the standard deviation, before processing. This often helps convergence.)

下面先把原文逐字不遗漏地译成中文，然后用通俗方式解释，最后给出要点小结。

翻译

图 11.11 把 EM 算法用于高斯混合模型 (GMM) 在 Old Faithful 数据上的示意。(a) 参数的初始 (随机) 取值。(b) 在第一次 E 步中为每个点计算得到的后验责任度。红色程度表示该点属于红色簇的程度，蓝色同理；紫色的点在各簇上的后验大致均匀。(c) 第一次 M 步之后更新的参数。(d) 第 3 次迭代后。(e) 第 5 次迭代后。(f) 第 16 次迭代后。基于 (Bishop 2006a) 图 9.8。图由 `mixGaussDemoFaithful` 生成。

11.4.2 GMM 的 EM 算法

本节讨论如何用 EM 拟合高斯混合。拟合其它类型的混合模型只需做直接的改动——见练习 11.3。我们假设混合分量个数 K 已知（关于这个问题的讨论见 11.5 节）。

11.4.2.1 辅助函数

期望的完备数据对数似然为

$$Q(\theta, \theta^{(t-1)}) = \mathbb{E} \left[\sum_i \log p(x_i, z_i | \theta) \right] \quad (11.22)$$

$$= \sum_i \mathbb{E} \left[\log \prod_{k=1}^K (\pi_k p(x_i | \theta_k))^{I(z_i=k)} \right] \quad (11.23)$$

$$= \sum_i \sum_k \mathbb{E}[I(z_i = k)] \log [\pi_k p(x_i | \theta_k)] \quad (11.24)$$

$$= \sum_i \sum_k p(z_i = k | x_i, \theta^{t-1}) \log [\pi_k p(x_i | \theta_k)] \quad (11.25)$$

$$= \sum_i \sum_k r_{ik} \log \pi_k + \sum_i \sum_k r_{ik} \log p(x_i | \theta_k) \quad (11.26)$$

其中 $r_{ik} = p(z_i = k | x_i, \theta^{(t-1)})$ 是第 k 个簇对第 i 个数据点的责任度。它在下面的 E 步中计算。

11.4.2.2 E 步

E 步形式很简单 (对任意混合模型都相同) :

$$r_{ik} = \frac{\pi_k p(x_i | \theta_k^{(t-1)})}{\sum_k \pi_k p(x_i | \theta_k^{(t-1)})} \quad (11.27)$$

11.4.2.3 M 步

在 M 步中, 对 π 与 θ_k 最大化 Q 。对 π 显然有

$$\pi_k = \frac{1}{N} \sum_i r_{ik} = \frac{r_k}{N} \quad (11.28)$$

其中 $r_k = \sum_i r_{ik}$ 是分配给簇 k 的**加权样本数**。

为了推出 μ_k 与 Σ_k 的 M 步, 看 Q 中依赖它们的部分。得到

$$(\mu_k, \Sigma_k) = \sum_k \sum_i r_{ik} \log p(x_i | \theta_k) \quad (11.29)$$

$$= -\frac{1}{2} \sum_i r_{ik} (\log |\Sigma_k| + (x_i - \mu_k)^\top \Sigma_k^{-1} (x_i - \mu_k)) \quad (11.30)$$

这就是**带权版本的“求多元高斯 MLE”**的标准问题 (见 4.1.3 节)。可证明 (练习 11.2) 新的参数估计为

$$\mu_k = \frac{\sum_i r_{ik} x_i}{r_k} \quad (11.31)$$

$$\Sigma_k = \frac{\sum_i r_{ik} (x_i - \mu_k)(x_i - \mu_k)^\top}{r_k} = \frac{\sum_i r_{ik} x_i x_i^\top}{r_k} - \mu_k \mu_k^\top \quad (11.32)$$

这些式子直观易懂: 簇 k 的均值就是分到该簇的点的**加权平均**, 协方差与**加权离差矩阵**成正比。计算出新估计后, 令

$\theta^t = (\pi_k, \mu_k, \Sigma_k)_{k=1:K}$, 然后进入下一次 E 步。

11.4.2.4 例子

算法运行的一个例子见图 11.11。我们从

$\mu_1 = (-1, 1)$, $\Sigma_1 = I$, $\mu_2 = (1, -1)$, $\Sigma_2 = I$ 开始。着色约定为: 蓝色点来自簇 1, 红色点来自簇 2。更准确地, 我们把颜色设为

$$\text{color}(i) = r_{i1} \cdot \text{blue} + r_{i2} \cdot \text{red} \quad (11.33)$$

因此不确定的点看起来是紫色。经过 20 次迭代, 算法收敛到一个不错的聚类。 (在处理前对数据做了**标准化**: 减去均值并除以标准差。这通常有助于收敛。)

通俗解释

1) 整体思路

- 在混合模型里，样本属于哪个簇是**未知的**。
- **E 步**: 用当前参数算“每个点属于每个簇的概率” r_{ik} （也叫“软标签/责任度”）。
- **M 步**: 把这些软标签当权重，重算每个簇的权重 π_k 、均值 μ_k 、协方差 Σ_k 。
- 循环往复，直到收敛（对数似然不再增长）。

2) 公式 (11.22)–(11.26) 的小窍门

- 用指示变量 $I(z_i = k)$ 把“属于第 k 簇”写进对数里；
- 对它求期望就得到

$$\mathbb{E}[I(z_i = k)] = p(z_i = k \mid x_i, \theta^{t-1}) = r_{ik}.$$
- 于是 Q 变成“每个簇对每个点的**加权和**”。

3) E 步的分母为什么是和？

- 分子是“先抽到簇 k ”×“在簇 k 下生成该点的可能性”；
- 对所有簇求和后归一化，就得到后验概率。

4) M 步的直觉

- π_k : 簇 k 的“市场份额” = 该簇吸到的**有效样本数** / 总样本数；
- μ_k : 该簇的“重心” = 按 r_{ik} 加权的均值；
- Σ_k : 该簇的“形状/大小” = 按 r_{ik} 加权的协方差（离散程度）。

5) 图 11.11 怎样读

- (a) 随机起步，两个椭圆（协方差等高线）随便放。
- (b) 第一次 E 步：点的颜色深浅体现 r_{ik} 。紫色=不确定。
- (c)(d)(e)(f) 随迭代推进，椭圆逐渐贴合两团数据，紫点减少，直到收敛。

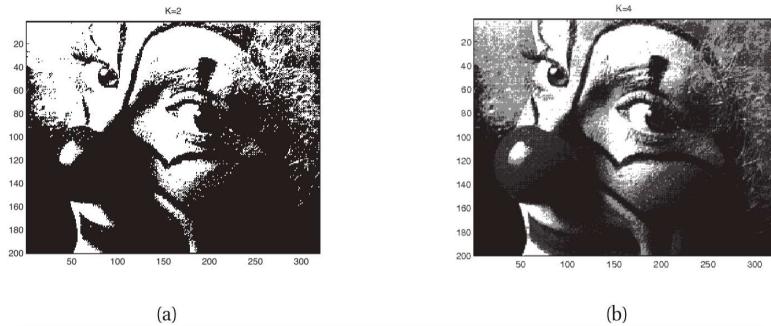
6) 实操小贴士

- **标准化**数据常能让收敛更稳。
- 多次随机初始化、选择对数似然最大的那次结果。
- 可用 K-means/++ 初始化 μ_k 与 π_k 来加速、稳定。

要点小结

- **E 步**: $r_{ik} = \frac{\pi_k p(x_i \mid \theta_k)}{\sum_j \pi_j p(x_i \mid \theta_j)}.$
- **M 步**:
 - $\pi_k = \frac{1}{N} \sum_i r_{ik};$

- $\mu_k = \frac{\sum_i r_{ik}x_i}{\sum_i r_{ik}}$;
- $\Sigma_k = \frac{\sum_i r_{ik}(x_i - \mu_k)(x_i - \mu_k)^\top}{\sum_i r_{ik}}$ 。
- **直觉：**E 步给“软分配”，M 步据此算“带权均值/协方差”；两步交替，似然单调上升，直到收敛。
- **示例：**在 Old Faithful 数据上，约 20 次迭代即可得到合理簇；颜色由责任度混合而成，紫色表示不确定。



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。

Figure 11.12 An image compressed using vector quantization with a codebook of size K. (a) K = 2. (b) K = 4. Figure generated by vqDemo.

11.4.2.5 K-means algorithm

There is a popular variant of the EM algorithm for GMMs known as the K-means algorithm, which we now discuss. Consider a GMM in which we make the following assumptions: $\Sigma_k = \sigma^2 I_D$ is fixed, and $\pi_k = 1/K$ is fixed, so only the cluster centers, $\mu_k \in \mathbb{R}^D$, have to be estimated.

Now consider the following delta-function approximation to the posterior computed during the E step:

$$p(z_i = k | x_i, \theta) \approx I(k = z^* i) \quad (11.34)$$

where $z^* = \text{argmax}_k p(z_i = k | x_i, \theta)$. This is sometimes called hard EM, since we are making a hard assignment of points to clusters. Since we assumed an equal spherical covariance matrix for each cluster, the most probable cluster for x_i can be computed by finding the nearest prototype:

$$z^* = \text{argmin}_k \|x_i - \mu_k\|^2 \quad (11.35)$$

Hence in each E step, we must find the Euclidean distance between N data points and K cluster centers, which takes O(NKD) time. However, this can be sped up using various techniques, such as applying the triangle inequality to avoid some redundant computations (Elkan 2003). Given the hard cluster assignments, the M step updates each cluster center by computing the mean of all points assigned to it:

$$\mu_k = \frac{1}{N_k} \sum_{i:z_i=k} x_i \quad (11.36)$$

11.4.2.6 Vector quantization

Since K-means is not a proper EM algorithm, it is not maximizing likelihood. Instead, it can be interpreted as a greedy algorithm for approximately minimizing a loss function related to data compression, as we now explain. Suppose we want to perform lossy compression of some real-valued vectors, $x_i \in \mathbb{R}^D$. A very simple approach to this is to use vector quantization or VQ. The basic idea is to replace each real-valued vector $x_i \in \mathbb{R}^D$ with a discrete symbol $z_i \in \{1, \dots, K\}$, which is an index into a codebook of K prototypes, $\mu_k \in \mathbb{R}^D$. Each data vector is encoded by using the index of the most similar prototype, where similarity is measured in terms of Euclidean distance:

$$\text{encode}(x_i) = \arg \min_k \|x_i - \mu_k\|^2 \quad (11.37)$$

We can define a cost function that measures the quality of a codebook by computing the reconstruction error or distortion it induces:

$$J(\mu, z|K, X) = \frac{1}{N} \sum_{i=1}^N \|x_i - \text{decode}(\text{encode}(x_i))\|^2 = \frac{1}{N} \sum_{i=1}^N \|x_i - \mu_{z_i}\|^2 \quad (11.38)$$

where $\text{decode}(k) = \mu_k$. The K-means algorithm can be thought of as a simple iterative scheme for minimizing this objective. Of course, we can achieve zero distortion if we assign one prototype to every data vector, but that takes $O(NDC)$ space, where N is the number of real-valued data vectors, each of length D, and C is the number of bits needed to represent a real-valued scalar (the quantization accuracy). However, in many data sets, we see similar vectors repeatedly, so rather than storing them many times, we can store them once and then create pointers to them. Hence we can reduce the space requirement to $O(N \log_2 K + KDC)$: the $O(N \log_2 K)$ term arises because each of the N data vectors needs to specify which of the K codewords it is using (the pointers); and the $O(KDC)$ term arises because we have to store each codebook entry,

each of which is a D-dimensional vector. Typically the first term dominates the second, so we can approximate the rate of the encoding scheme (number of bits needed per object) as $O(\log_2 K)$, which is typically much less than $O(DC)$.

One application of VQ is to image compression. Consider the $N = 200 \times 320 = 64,000$ pixel image in Figure 11.12; this is gray-scale, so $D = 1$. If we use one byte to represent each pixel (a gray-scale intensity of 0 to 255), then $C = 8$, so we need $NC = 512,000$ bits to represent the image. For the compressed image, we need $N \log_2 K + KC$ bits. For $K = 4$, this is about 128kb, a factor of 4 compression. For $K = 8$, this is about 192kb, a factor of 2.6 compression, at negligible perceptual loss (see Figure 11.12(b)). Greater compression could be achieved if we modelled spatial correlation between the pixels, e.g., if we encoded 5x5 blocks (as used by JPEG). This is because the residual errors (differences from the model's predictions) would be smaller, and would take fewer bits to encode.

11.4.2.7 Initialization and avoiding local minima

Both K-means and EM need to be initialized. It is common to pick K data points at random, and to make these be the initial cluster centers. Or we can pick the centers sequentially so as to try to "cover" the data. That is, we pick the initial point uniformly at random. Then each subsequent point is picked from the remaining points with probability proportional to its squared distance to the point's closest cluster center. This is known as farthest point clustering (Gonzales 1985), or k-means++ (Arthur and Vassilvitskii 2007; Bahmani et al. 2012).

Surprisingly, this simple trick can be shown to guarantee that the distortion is never more than $O(\log K)$ worse than optimal (Arthur and Vassilvitskii 2007).

An heuristic that is commonly used in the speech recognition community is to incrementally "grow" GMMs: we initially give each cluster a score based on its mixture weight; after each round of training, we consider splitting the cluster with the highest score into two, with the new centroids being random perturbations of the original centroid, and the new scores being half of the old scores. If a new cluster has too small a score, or too narrow a variance, it is removed. We continue in this way until the

desired number of clusters is reached. See (Figueiredo and Jain 2002) for a similar incremental approach.

已思考 6s

下面先把原文完整翻译（不遗漏），再用通俗解释，最后给出要点小结。

翻译

图 11.12 用码本大小为 K 的**向量量化** (vector quantization, VQ) 对图像进行压缩的结果。 (a) $K = 2$; (b) $K = 4$ 。图由 `vqDemo` 生成。

11.4.2.5 K-means 算法

在 GMM 的 EM 算法中有一个很流行的变体，称为 **K-means** 算法，下面加以讨论。考虑一个高斯混合模型，并作如下假设： $\Sigma_k = \sigma^2 I_D$ 固定，且 $\pi_k = 1/K$ 固定，因此只需要估计**簇中心** $\mu_k \in \mathbb{R}^D$ 。再看一下在 E 步计算的后验的下面这种**δ 函数近似**：

$$p(z_i = k | x_i, \theta) \approx \mathbf{1}(k = z_i^{*}) \quad (11.34)$$

其中 $z_i^{*} = \arg \max_k p(z_i = k | x_i, \theta)$ 。这有时称为 **hard EM**，因为我们把样本**硬性分配**给某个簇。由于我们假定每个簇的协方差都是相同的球形矩阵， x_i 最可能的簇可以通过寻找**最近的原型**得到：

$$z_i^{*} = \arg \min_k \|x_i - \mu_k\|_2^2 \quad (11.35)$$

因此，每次 E 步都要计算 N 个数据点到 K 个簇中心的欧氏距离，耗时 $O(NKD)$ 。不过可以用一些技巧加速，比如用**三角不等式**跳过部分冗余计算 (Elkan 2003)。给定硬分配后的簇标签，M 步把每个簇中心更新为分给该簇的点的均值：

$$\mu_k = \frac{1}{N_k} \sum_{i:z_i=k} x_i. \quad (11.36)$$

11.4.2.6 向量量化

由于 K-means 不是一个真正的 EM 算法，它**并不最大化似然**。相反，它可以理解为一种贪心算法，用来**近似最小化与数据压缩相关的损失函数**，下面加以说明。假设我们要对一些实值向量 $x_i \in \mathbb{R}^D$ 做**有损压缩**。一种非常简单的方法是用**向量量化** (VQ)。基本思想是：把每个实值向量 x_i 替换为一个离散符号 $z_i \in \{1, \dots, K\}$ ，它是一个指向含 K 个原

型 $\mu_k \in \mathbb{R}^D$ 的码本的索引。每个数据向量通过与最相似的原型的索引来编码，相似度用欧氏距离衡量：

$$\text{encode}(x_i) = \arg \min_k \|x_i - \mu_k\|^2. \quad (11.37)$$

我们可以用**重建误差/失真**来度量一个码本的好坏：

$$J(\mu, z | K, X) = \frac{1}{N} \sum_{i=1}^N \|x_i - \text{decode}(\text{encode}(x_i))\|^2 = \frac{1}{N} \sum_{i=1}^N \|x_i - \mu_{z_i}\|^2 \quad (11.38)$$

其中 $\text{decode}(k) = \mu_k$ 。K-means 可以看作是一个**迭代地最小化**该目标的简单方案。

当然，如果给每个数据向量都分配一个独立原型，失真可以为 0，但这需要 $O(NDC)$ 的存储，其中 N 是向量个数、每个向量维度为 D 、 C 是表示一个实数所需的比特数（量化精度）。然而在许多数据集中，常会重复出现相似的向量，因此我们可以只把它们存一次，然后用**指针**指向它们。于是存储可降为

$$O(N \log_2 K + KDC) :$$

其中 $O(N \log_2 K)$ 是因为每个数据向量要说明它使用了 K 个码字中的哪一个（指针）； $O(KDC)$ 是因为要存码本中每个 D 维的原型。通常第一项占主导，因此每个对象所需的编码比特数（码率）近似为 $O(\log_2 K)$ ，这通常远小于 $O(DC)$ 。

VQ 的一个应用是**图像压缩**。看图 11.12 的图像，像素数 $N = 200 \times 320 = 64,000$ ，灰度图，故 $D = 1$ 。如果每个像素用 1 字节表示（灰度 0–255），则 $C = 8$ ，因此原图需要 $NC = 512,000$ 比特。压缩后需要 $N \log_2 K + KC$ 比特。若 $K = 4$ ，约为 128kb，压缩比 4；若 $K = 8$ ，约为 192kb，压缩比 2.6，且几乎无可觉察的损失（见图 11.12(b)）。如果进一步建模像素间的**空间相关**（例如把 5×5 的块作为单位——JPEG 的做法），还可以获得更高的压缩比，因为**残差**（与模型预测的差）更小，用更少比特即可编码。

11.4.2.7 初始话与避免局部极小

K-means 与 EM 都需要初始化。常见做法是**随机选取** K 个数据点作为初始中心；或者**顺序挑选**中心以“覆盖”数据：先均匀随机选第一个点，然后每次从剩余点中按其到最近中心的**平方距离**成比例抽取下一个。这称为**最远点聚类**（Gonzales 1985），又叫 k-means++（Arthur & Vassilvitskii 2007；Bahmani 等 2012）。令人惊讶的是，可证明这种简单技巧能保证失真最多只比最优值差一个 $O(\log K)$ 的因子（Arthur & Vassilvitskii 2007）。

语音识别领域常用的一个启发式是**增量生长 GMM**：起初按混合权重给每个簇打分；每轮训练后，把分数最高的簇**一分为二**，新中心是原中心的随机微扰，新权重是原权重的一

半；若新簇的分数太小或方差过窄，则删除之。如此继续，直到达到期望的簇数。类似的增量方法可参见 Figueiredo & Jain (2002)。

通俗解释

1) K-means和EM/GMM的关系

- 假设所有簇的形状都一样（球形，方差相同），权重也相同；只需要找各簇的“中心”。
- Hard EM**: 在 E 步里，不算“属于每个簇的概率”，而是直接挑一个最近的簇（最近中心）。
- M 步**: 把该簇的中心更新为分到它的样本的**平均值**。
- 如此反复，直到簇中心不再变化或失真下降很少。

2) 为什么说 K-means \neq 最大似然？

- 它不再使用 GMM 的完整概率模型（固定 Σ, π ，且把后验硬化），因此优化的不是似然，而是**重建误差**
$$\frac{1}{N} \sum \|x_i - \mu_{z_i}\|^2.$$
- 这恰好对应**向量量化**中的“码本质量”。

3) VQ 的存储账本

- 原始：每个像素/向量都要存全部数值 \rightarrow 约 $O(DC)$ 比特/对象。
- VQ：只存一个**码本** (K 个 prototype)，再给每个对象存一个**索引** ($\log_2 K$ 比特) \rightarrow 总存储 $N \log_2 K + KDC$ 。
- 直观：常见的模式只存一次，其他地方用“指针”引用。

4) 读图 11.12

- $K = 2$ ：只有黑/白两类，块感强、失真大。
- $K = 4$ ：灰阶更丰富，视觉效果明显提升；仍然比分别存 256 个灰度值节省大量比特。

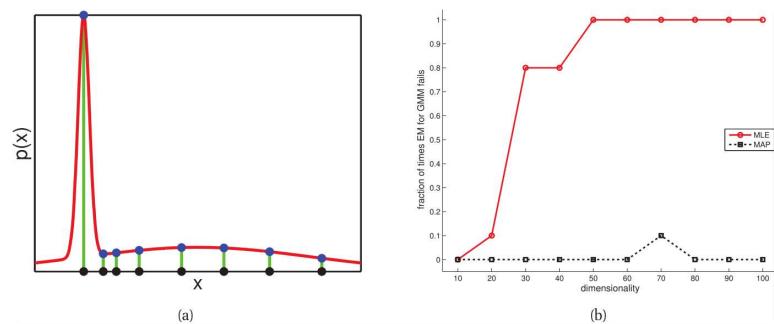
5) 如何避免“坏初始化”

- k-means++**：优先选择离已选中心远的点作新中心 \rightarrow 更稳定、更快收敛，还有近似最优保证。
 - 增量 GMM**：按权重与方差分裂/删除簇，逐步长到需要的 K 。
-

要点小结

- K-means = Hard-EM(GMM + 球协方差 + 等权重)**：

- E 步：把每个点分到最近中心；
- M 步：中心取该簇样本的**均值**；
- 计算量 $O(NKD)$, 可用三角不等式等加速。
- **目标等价**: 最小化**重建误差/失真**, 与 VQ 的码本优化一致; 不是最大似然。
- **VQ 压缩存储**: 总比特约 $N \log_2 K + KDC$; 通常主项是 $N \log_2 K \rightarrow$ 每对象约 $\log_2 K$ 比特。
- **图像压缩例子**: K 越大, 压缩率下降但画质更好; 结合空间块 (如 JPEG) 还能进一步压缩。
- **初始化与局部最小**: 用 k-means++ 明显更稳; GMM 可用增量分裂/删除簇的启发式逐步“长大”。



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。

Figure 11.13 (a) Illustration of how singularities can arise in the likelihood function of GMMs. Based on (Bishop 2006a) Figure 9.7. Figure generated by mixGaussSingularity. (b) Illustration of the benefit of MAP estimation vs ML estimation when fitting a Gaussian mixture model. We plot the fraction of times (out of 5 random trials) each method encounters numerical problems vs the dimensionality of the problem, for $N = 100$ samples. Solid red (upper curve): MLE. Dotted black (lower curve): MAP. Figure generated by mixGaussMLvsMAP.

11.4.2.8 MAP estimation

As usual, the MLE may overfit. The overfitting problem is particularly severe in the case of GMMs. To understand the problem, suppose for simplicity that $\Sigma_k = \sigma^2 I$, and

that $K = 2$. It is possible to get an infinite likelihood by assigning one of the centers, say μ_2 , to a single data point, say x_1 , since then the 1st term makes the following contribution to the likelihood:

$$N(x_1 | \mu_2, \sigma^2) = 1 / 2\pi\sigma^2 e^{-\frac{1}{2}\sigma^2} (11.39)$$

Hence we can drive this term to infinity by letting $\sigma^2 \rightarrow 0$, as shown in Figure 11.13(a). We will call this the “collapsing variance problem”. An easy solution to this is to perform MAP estimation. The new auxiliary function is the expected complete data log-likelihood plus the log prior:

$$Q(\theta, \theta_{old}) = \sum_k r_{ik} \log \pi_k + \sum_i r_{ik} \log p(x_i | \theta_k) + \log p(\pi) + k \log p(\theta_k) (11.40)$$

Note that the E step remains unchanged, but the M step needs to be modified, as we now explain. For the prior on the mixture weights, it is natural to use a Dirichlet prior, $\pi \sim \text{Dir}(\alpha)$, since this is conjugate to the categorical distribution. The MAP estimate is given by

$$\pi_k = r_k + \alpha_k - 1 / N + k \alpha_k - K (11.41)$$

If we use a uniform prior, $\alpha_k = 1$, this reduces to Equation 11.28. The prior on the parameters of the class conditional densities, $p(\theta_k)$, depends on the form of the class conditional densities. We discuss the case of GMMs below, and leave MAP estimation for mixtures of Bernoullis to Exercise 11.3. For simplicity, let us consider a conjugate prior of the form

$$p(\mu_k, \Sigma_k) = \text{NIW}(\mu_k, \Sigma_k | m_0, \kappa_0, v_0, S_0) (11.42)$$

From Section 4.6.3, the MAP estimate is given by

$$\hat{\mu}_k = \frac{r_k x_k + \kappa_0 m_0}{r_k + \kappa_0} (11.43) (11.44)$$

$$x_k = \frac{1}{r_k} \sum_i r_{ik} x_i (11.45)$$

$$\begin{aligned} \hat{\Sigma}_k &= S_0 + \sum_i r_{ik} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T \\ &+ D^{-1} (11.46) \end{aligned}$$

$$S_k = \sum_i r_{ik} (x_i - x_k)(x_i - x_k)^T (11.47)$$

We now illustrate the benefits of using MAP estimation instead of ML estimation in the context of GMMs. We apply EM to some synthetic data in D dimensions, using either ML or MAP estimation. We count the trial as a “failure” if there are numerical issues involving singular matrices. For each dimensionality, we conduct 5 random trials. The results are illustrated in Figure 11.13(b) using $N = 100$. We see that as soon as D becomes even moderately large, ML estimation crashes and burns, whereas MAP estimation never encounters numerical

problems. When using MAP estimation, we need to specify the hyper-parameters. Here we mention some simple heuristics for setting them (Fraley and Raftery 2007, p163). We can set $\kappa_0 = 0$, so that the μ_k are unregularized, since the numerical problems only arise from Σ_k . In this case, the MAP estimates simplify to $\hat{\mu}_k = \bar{x}_k$ and $\hat{\Sigma}_k = S_0 + S_k v_0 + r_k + D + 2$, which is not quite so scary-looking. Now we discuss how to set S_0 . One possibility is to use

$$S_0 = 1/K_1/D \text{ diag}(s_2^2 1, \dots, s_D^2 D) \quad (11.48)$$

where $s_j = (1/N) \sum_{i=1}^N (x_{ij} - \bar{x}_j)^2$ is the pooled variance for dimension j . (The reason for the $1/K_1/D$ term is that the resulting volume of each ellipsoid is then given by $|S_0| = 1/K_1 | \text{diag}(s_2^2 1, \dots, s_D^2 D) |$.) The parameter v_0 controls how strongly we believe this prior. The weakest prior we can use, while still being proper, is to set $v_0 = D + 2$, so this is a common choice.

下面先把原文“逐字不遗漏”地译成中文，然后用通俗解释，最后给出要点小结。

翻译

图 11.13 (a) 说明在 GMM 的似然函数中如何产生奇异性。基于 (Bishop 2006a) 图 9.7。图由 `mixGaussSingularity` 生成。(b) 说明在拟合高斯混合模型时，用 MAP 估计相对 ML 估计的好处。我们在样本量 $N = 100$ 的条件下，按照问题维度作图，纵轴为两种方法在 5 次随机试验中遇到数值问题的比例。红色实线（上方曲线）：MLE。黑色虚线（下方曲线）：MAP。图由 `mixGaussMLvsMAP` 生成。

11.4.2.8 MAP 估计

像往常一样，MLE 可能过拟合。对 GMM 来说，这个问题尤其严重。为理解它，出于简便起见，设 $\Sigma_k = \sigma_k^2 I$ 且 $K = 2$ 。通过把某个中心（如 μ_2 ）对准某个数据点（如 x_1 ），可以使似然无穷大，因为第一项对似然的贡献是

$$N(x_1 | \mu_2, \sigma_2^2 I) = \frac{1}{(2\pi\sigma_2^2)^{D/2}} e^0 \quad (11.39)$$

于是让 $\sigma_2 \rightarrow 0$ 就能把这一项推到无穷大，如图 11.13(a) 所示。我们把这叫作“方差塌缩问题”。

一个简单的解决办法是改用 MAP 估计。新的辅助函数（期望的完备数据对数似然 + 对数先验）为

$$Q(\theta, \theta_{\text{old}}) = \sum_{i,k} r_{ik} \log \pi_k + \sum_{i,k} r_{ik} \log p(x_i | \theta_k) + \log p(\boldsymbol{\pi}) + \sum_k \log p(\theta_k) \quad (11.40)$$

注意 E 步不变，但 M 步需要修改。对混合权重的先验，自然选 Dirichlet： $\boldsymbol{\pi} \sim \text{Dir}(\boldsymbol{\alpha})$ ，因为它与 Categorical 共轭。其 MAP 为

$$\pi_k = \frac{r_k + \alpha_k - 1}{N + \sum_j \alpha_j - K} \quad (11.41)$$

若用均匀先验 $\alpha_k = 1$ ，就退化为式 (11.28)。

类条件参数的先验 $p(\theta_k)$ 取决于类条件分布的形式。下面讨论 GMM 的情形（伯努利混合见练习 11.3）。为简明起见，取共轭先验

$$p(\mu_k, \Sigma_k) = \text{NIW}(\mu_k, \Sigma_k | m_0, \kappa_0, v_0, S_0) \quad (11.42)$$

由 4.6.3 节，MAP 估计为

$$\hat{\mu}_k = \frac{r_k \bar{x}_k + \kappa_0 m_0}{r_k + \kappa_0}, \quad \bar{x}_k = \frac{\sum_i r_{ik} x_i}{r_k} \quad (11.43-11.45)$$

$$\hat{\Sigma}_k = \frac{S_0 + S_k + \frac{\kappa_0 r_k}{\kappa_0 + r_k} (\bar{x}_k - m_0)(\bar{x}_k - m_0)^\top}{v_0 + r_k + D + 2} \quad (11.46)$$

$$S_k = \sum_i r_{ik} (x_i - \bar{x}_k)(x_i - \bar{x}_k)^\top. \quad (11.47)$$

下面通过 GMM 的一个例子说明 MAP 相对 ML 的优点。我们在维度为 D 的合成数据上，用 EM 分别做 ML 与 MAP；若出现奇异矩阵等数值问题，就把该次试验计为“失败”。对每个维度做 5 次随机试验。图 11.13(b) 展示了 $N = 100$ 时的结果：只要 D 稍微大一些，ML 就频繁“崩溃”，而 MAP 几乎从不出数值问题。

使用 MAP 时需要指定超参数。下面给出一些简单的启发式 (Fraley & Raftery 2007, p163)。可以取 $\kappa_0 = 0$ ，即对 μ_k 不作正则化，因为数值问题只来自 Σ_k 。此时

$$\hat{\mu}_k = \bar{x}_k, \quad \hat{\Sigma}_k = \frac{S_0 + S_k}{v_0 + r_k + D + 2},$$

形式更直观。如何设 S_0 ？一种做法是

$$S_0 = \frac{1}{K^{1/D}} \text{diag}(s_1^2, \dots, s_D^2) \quad (11.48)$$

其中 $s_j^2 = \frac{1}{N} \sum_{i=1}^N (x_{ij} - \bar{x}_j)^2$ 是第 j 维的合并方差。（系数 $K^{-1/D}$ 的原因是这样得到的每个椭球体积为 $|S_0| = \frac{1}{K} |\text{diag}(s_1^2, \dots, s_D^2)|$ 。）参数 v_0 控制先验强度；在保证先验“合

适”的前提下最弱的选择是 $\nu_0 = D + 2$, 这也是常见设定。

通俗解释

1) 为什么 ML 会“炸”?

- 把某个高斯分量的均值放到某个样本上, 再把该分量的方差收缩到 0, 密度会趋向无穷大 → 似然无穷大 → 算法走向**方差塌缩**。这既是**过拟合**也是**数值奇异** (协方差不可逆)。

2) MAP 怎么救?

- 给参数加**先验** (π 用 Dirichlet, μ, Σ 用 NIW)。
- M 步里多了先验项, 相当于对极端的解“拉回中庸”, 阻止方差塌成 0。
- 权重公式 (11.41) 是把“软计数” r_k 与先验“伪计数” $\alpha_k - 1$ 相加;
- $\hat{\mu}, \hat{\Sigma}$ 是“数据统计量 + 先验统计量”的折中。

3) 图 11.13(b) 的信息

- 在同样的样本量 $N = 100$ 下, 维度一上去, **MLE 几乎必挂**;
- MAP 基本不出数值问题 → 实战中更稳。

4) 超参数怎么设 (实操版)

- 取 $\kappa_0 = 0$: 只管协方差, 均值不正则。
 - 取 $\nu_0 = D + 2$: 最弱但合格的先验强度。
 - 取 $S_0 = K^{-1/D} \text{diag}(s_j^2)$: 让每个初始协方差椭球的体积约为整体方差体积的 $1/K$ 。
-

要点小结

- **ML 在 GMM 中会出现“方差塌缩”→ 似然无穷大 → 数值奇异。**
- **MAP = ML + 先验**: 在 M 步最大化 $Q + \log p(\theta)$, 可有效抑制塌缩。
- 具体 MAP 更新 (NIW 先验):
 - $\pi_k = \frac{r_k + \alpha_k - 1}{N + \sum_j \alpha_j - K};$
 - $\hat{\mu}_k = \frac{r_k \bar{x}_k + \kappa_0 m_0}{r_k + \kappa_0};$
 - $\hat{\Sigma}_k = \frac{S_0 + S_k + \frac{\kappa_0 r_k}{\kappa_0 + r_k} (\bar{x}_k - m_0)(\bar{x}_k - m_0)^\top}{\nu_0 + r_k + D + 2}.$
- **经验设定**: $\kappa_0 = 0$ 、 $\nu_0 = D + 2$ 、 $S_0 = K^{-1/D} \text{diag}(s_j^2)$ 。
- **结论**: 在 GMM 上, MAP 比 ML 稳健得多, 尤其在中高维或样本不多时, 能避免奇异与过拟合。

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。

11.4.3 EM for mixture of experts

We can fit a mixture of experts model using EM in a straightforward manner. The expected complete data log likelihood is given by

$$Q(\theta, \theta_{old}) = \sum_{i=1}^N \sum_{k=1}^K r_{ik} \log[\pi_k N(y_i | w_k^T x_i, \sigma_k^2)] \quad (11.49)$$

$$\pi_{i,k} = S(Vx_i)_k \quad (11.50)$$

$$r_{ik} \propto \pi_{old,k} N(y_i | x_i, \sigma_{old,k}^2) \quad (11.51)$$

So the E step is the same as in a standard mixture model, except we have to replace π_k with $\pi_{i,k}$ when computing r_{ik} .

In the M step, we need to maximize $Q(\theta, \theta_{old})$ wrt w_k , σ_k^2 and V . For the regression parameters for model k , the objective has the form

$$Q(\theta_k, \theta_{old,k}) = \sum_{i=1}^N r_{ik} (y_i - w_k^T x_i)^2 \quad (11.52)$$

We recognize this as a weighted least squares problem, which makes intuitive sense: if r_{ik} is small, then data point i will be downweighted when estimating model k 's parameters. From Section 8.3.4 we can immediately write down the MLE as

$$w_k = (X^T R_k X)^{-1} X^T R_k y \quad (11.53)$$

where $R_k = \text{diag}(r_{:,k})$. The MLE for the variance is given by

$$\sigma_k^2 = \frac{1}{N} \sum_{i=1}^N r_{ik} (y_i - w_k^T x_i)^2 \quad (11.54)$$

We replace the estimate of the unconditional mixing weights π with the estimate of the gating parameters, V . The objective has the form

$$I(V) = \sum_{i=1}^N \sum_{k=1}^K r_{ik} \log \pi_{i,k} \quad (11.55)$$

We recognize this as equivalent to the log-likelihood for multinomial logistic regression in Equation 8.34, except we replace the "hard" 1-of-C encoding y_i with the "soft" 1-of-K encoding r_i . Thus we can estimate V by fitting a logistic regression model to soft target labels.

11.4.4 EM for DGMs with hidden variables

We can generalize the ideas behind EM for mixtures of experts to compute the MLE or MAP estimate for an arbitrary DGM. We could use gradient-based methods

(Binder et al. 1997), but it is much simpler to use EM (Lauritzen 1995): in the E step, we just estimate the hidden variables, and in the M step, we will compute the MLE using these filled-in values. We give the details below. For simplicity of presentation, we will assume all CPDs are tabular. Based on Section 10.4.2, let us write each CPT as follows:

$$p(x_{it}|x_i, pa(t), \theta_t) = K_{pa(t)} c=1 K_t k=1 \theta_l(x_{it}=i, x_i, pa(t)=c) t_{ck} \quad (11.56)$$

The log-likelihood of the complete data is given by
 $\log p(D|\theta) = \sum_{t=1}^T \sum_{c=1}^{K_{pa(t)}} \sum_{k=1}^{N_{tck}} \log \theta_{tck}$ (11.57)
 where $N_{tck} = \sum_{i=1}^N I(x_{it} = i, x_i, pa(t) = c)$ are the empirical counts. Hence the expected complete data log-likelihood has the form

$$E[\log p(D|\theta)] = \sum_{t=1}^T \sum_{c=1}^{K_{pa(t)}} N_{tck} \log \theta_{tck} \quad (11.58)$$

where

$$N_{tck} = \sum_{i=1}^N E[I(x_{it} = i, x_i, pa(t) = c)] = \sum_{i=1}^N p(x_{it} = c | D_i) \quad (11.59)$$

where D_i are all the visible variables in case i . The quantity $p(x_{it}, x_i, pa(t)|D_i, \theta)$ is known as a family marginal, and can be computed using any GM inference algorithm. The N_{tjk} are the expected sufficient statistics, and constitute the output of the E step. Given these ESS, the M step has the simple form

$$\hat{\theta}_{tck} = \frac{N_{tck}}{N} \quad (11.60)$$

This can be proved by adding Lagrange multipliers (to enforce the constraint $\sum_k \hat{\theta}_{tjk} = 1$) to the expected complete data log likelihood, and then optimizing each parameter vector θ_{tc} separately. We can modify this to perform MAP estimation with a Dirichlet prior by simply adding pseudo counts to the expected counts.

下面先把原文逐句不遗漏地译成中文，然后给出通俗解释，最后总结要点。

翻译

11.4.3 专家混合 (Mixture of Experts, MoE) 的 EM

我们可以用 EM 以直接的方式拟合一个专家混合模型。期望的完备数据对数似然为

$$Q(\theta, \theta_{\text{old}}) = \sum_{i=1}^N \sum_{k=1}^K r_{ik} \log [\pi_{i,k} N(y_i | w_k^\top x_i, \sigma_k^2)] \quad (11.49)$$

$$\pi_{i,k} = S(V^\top x_i)_k \quad (11.50)$$

$$r_{ik} \propto \pi_{i,k}^{\text{old}} N(y_i | (x_i)^\top w_k^{\text{old}}, (\sigma_k^{\text{old}})^2) \quad (11.51)$$

因此，E 步与标准混合模型相同，只是在计算 r_{ik} 时要把 π_k 替换为 **依赖输入的** $\pi_{i,k}$ 。

在 M 步中，我们需要对 w_k , σ_k^2 和 V 最大化 $Q(\theta, \theta_{\text{old}})$ 。对第 k 个子模型的回归参数，目标函数为

$$Q(\theta_k, \theta_{\text{old}}) = \sum_{i=1}^N r_{ik} \left[-\frac{1}{\sigma_k^2} (y_i - w_k^\top x_i)^2 \right] \quad (11.52)$$

我们可将其识别为**加权最小二乘**问题；这也很直观：若 r_{ik} 小，则在估计模型 k 的参数时，第 i 个样本的权重就小。根据 8.3.4 节，可直接写出 MLE：

$$w_k = (X^\top R_k X)^{-1} X^\top R_k y \quad (11.53)$$

其中 $R_k = \text{diag}(r_{:,k})$ 。方差的 MLE 为

$$\sigma_k^2 = \frac{\sum_i r_{ik} (y_i - w_k^\top x_i)^2}{\sum_i r_{ik}}. \quad (11.54)$$

我们把“无条件的”混合权重 π 的估计，替换成**门控参数** V 的估计。其目标函数为

$$\ell(V) = \sum_i \sum_k r_{ik} \log \pi_{i,k}. \quad (11.55)$$

这等价于多项逻辑回归（式 8.34）的对数似然，只是把“硬”的 one-of- C 编码 y_i 换成了“软”的 one-of- K 编码 r_i 。因此我们可以对“软标签”做一次逻辑回归来估计 V 。

11.4.4 含隐藏变量的有向图模型 (DGM) 的 EM

我们可以把“用于专家混合的 EM”的思想推广到任意 DGM 上，以计算 MLE 或 MAP。虽然也可以用基于梯度的方法（Binder 等，1997），但用 EM 更简单（Lauritzen，1995）：在 E 步估计隐藏变量，在 M 步用这些“填充过”的值来计算 MLE。下面给出细节。为简化表述，假设所有条件概率分布 (CPD) 都是**表格型**。

根据 10.4.2 节，把每个条件概率表 (CPT) 写作

$$p(x_t^i | x_{pa(t)}^i, \theta_t) = \prod_{c=1}^{K_{pa(t)}} \prod_{k=1}^{K_t} \theta_{tck}^{\mathbf{1}(x_t^i = k, x_{pa(t)}^i = c)} \quad (11.56)$$

完备数据的对数似然为

$$\log p(\mathbf{D} \mid \theta) = \sum_{t=1}^T \sum_{c=1}^{K_{pa(t)}} \sum_{k=1}^{K_t} N_{tck} \log \theta_{tck} \quad (11.57)$$

其中

$$N_{tck} = \sum_{i=1}^N \mathbf{1}(x_t^i = k, x_{pa(t)}^i = c)$$

是经验计数。于是期望的完备数据对数似然为

$$E[\log p(\mathbf{D} \mid \theta)] = \sum_{t,c,k} \bar{N}_{tck} \log \theta_{tck} \quad (11.58)$$

其中

$$\bar{N}_{tck} = \sum_{i=1}^N E[\mathbf{1}(x_t^i = k, x_{pa(t)}^i = c)] = \sum_i p(x_t^i = k, x_{pa(t)}^i = c \mid D_i) \quad (11.59)$$

D_i 是第 i 个样本中所有可见变量。量

$p(x_t, x_{pa(t)} \mid D_i, \theta)$ 称为一个家族边缘分布 (family marginal)，可用任意图模型推断算法求得。 \bar{N}_{tck} 是期望充分统计量 (ESS)，构成 E 步的输出。给定这些 ESS，M 步的形式很简单：

$$\hat{\theta}_{tck} = \frac{\bar{N}_{tck}}{\sum_j \bar{N}_{tcj}}. \quad (11.60)$$

这可通过在期望对数似然上加入拉格朗日乘子 (约束 $\sum_k \theta_{tck} = 1$) 并分别优化每个参数向量 θ_{tc} 得到。若要做 MAP，只需在期望计数上加伪计数 (相当于给 θ_{tc} 放置 Dirichlet 先验)。

通俗解释

1) 用 EM 拟合 专家混合 (MoE)

- **模型**: 多个“专家”各自做回归；一个“门控函数”根据输入 x_i 给每个专家一个权重 $\pi_{i,k}$ (softmax 输出)。
- **E 步**: 像普通混合一样算“责任度” r_{ik} :
“门控权重 \times 专家在该点的似然”，再在所有专家上归一化。
- **M 步**:
 - 对每个专家 k : 解加权最小二乘得到 w_k ，用加权残差算 σ_k^2 。

- 对门控参数 V : 把 r_{ik} 当作“软标签”，做一次多项逻辑回归来学习 V 。

直觉：哪个专家在某个 x_i 上更靠谱 ($\pi_{i,k}$ 大、残差小)，它就获得更高的 r_{ik} 与更大的权重去更新自己的参数。

2) EM 处理**任意 DGM (有向图模型) **的套路

- E 步：在每个样本 i 上，用图模型推断算出每个 CPT 家族 $(x_t, pa(t))$ 取到各个状态的后验概率；把它当成“软计数” \bar{N}_{tck} 。
- M 步：每个 CPT 的参数就是“行内归一化的软计数”：

$$\hat{\theta}_{tck} = \bar{N}_{tck} / \sum_j \bar{N}_{tcj}.$$
- 做 MAP：给每个 \bar{N}_{tck} 加上 Dirichlet 先验的伪计数即可。

这与我们在“完全观测”情形下的极大似然是一致的：只是把真实计数换成了期望计数 (ESS)。

要点小结

- MoE 的 EM
 - E: $r_{ik} \propto \pi_{i,k} N(y_i | w_k^\top x_i, \sigma_k^2)$;
 - M:
 - $w_k = (X^\top R_k X)^{-1} X^\top R_k y$,
 - $\sigma_k^2 = \sum r_{ik} (y_i - w_k^\top x_i)^2 / \sum r_{ik}$,
 - V : 对“软标签” r_{ik} 做多项逻辑回归最大化 $\sum r_{ik} \log \pi_{i,k}$ 。
- DGM 的 EM
 - E: 对每个 CPT 家族算后验联合 $p(x_t, pa(t) | D_i) \rightarrow$ 得到期望计数 \bar{N}_{tck} ;
 - M: 参数 = 期望计数归一化, $\hat{\theta}_{tck} = \bar{N}_{tck} / \sum_j \bar{N}_{tcj}$;
 - MAP: 在计数上加 Dirichlet 伪计数。
- 核心直觉：把看不见的“硬计数/硬标签”，换成用当前参数算出来的“软计数/软标签”，再按完全数据的公式更新参数。这样就把“难的 $\log \sum$ ”变成了“易的求和 + 极大化”。

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。

11.4.5 EM for the Student distribution

One problem with the Gaussian distribution is that it is

sensitive to outliers, since the logprobability only decays quadratically with distance from the center. A more robust alternative is the Student t distribution, as discussed in Section ???. Unlike the case of a Gaussian, there is no closed form formula for the MLE of a Student, even if we have no missing data, so we must resort to iterative optimization methods. The easiest one to use is EM, since it automatically enforces the constraints that v is positive and that Σ is symmetric positive definite. In addition, the resulting algorithm turns out to have a simple intuitive form, as we see below. At first blush, it might not be apparent why EM can be used, since there is no missing data. The key idea is to introduce an “artificial” hidden or auxiliary variable in order to simplify the algorithm. In particular, we will exploit the fact that a Student distribution can be written as a Gaussian scale mixture:

$$T(x_i|\mu, \Sigma, v) = N(x_i|\mu, \Sigma/v_i) \text{Ga}(v_i|v/2, v/2) dz_i \quad (11.61)$$

(See Exercise 11.1 for a proof of this in the 1d case.) This can be thought of as an “infinite” mixture of Gaussians, each one with a slightly different covariance matrix.

Treating the z_i as missing data, we can write the complete data log likelihood as

$$l_c(\theta) = \sum_{i=1}^N [\log N(x_i|\mu, \Sigma/z_i) + \log \text{Ga}(z_i|v/2, v/2)] \quad (11.62)$$

$$= N \sum_{i=1}^N -\frac{D}{2} \log(2\pi) - \frac{1}{2} \log |\Sigma| - z_i^T \delta_i + \frac{v}{2} \log v - \log \Gamma(v/2) \quad (11.63)$$

$$+ \frac{v}{2} (\log z_i - \log \delta_i) + (D/2 - 1) \log z_i \quad (11.64)$$

where we have defined the Mahalanobis distance to be
 $\delta_i = (x_i - \mu)^T \Sigma^{-1} (x_i - \mu) \quad (11.65)$

We can partition this into two terms, one involving μ and Σ , and the other involving v . We have, dropping irrelevant constants,

$$c(\theta) = LN(\mu, \Sigma) + LG(v) \quad (11.66)$$

$$LN(\mu, \Sigma) = -\frac{1}{2} N \log |\Sigma| - \frac{1}{2} N \sum_{i=1}^N z_i^T \delta_i \quad (11.67)$$

$$LG(v) = -N \log \Gamma(v/2) + \frac{1}{2} N v \log(v/2) + \frac{1}{2} v N \sum_{i=1}^N (\log z_i - \log \delta_i) \quad (11.68)$$

下面先把原文完整翻译（不遗漏），然后给出通俗解释，最后做要点小结。

翻译

11.4.5 Student 分布的 EM

高斯分布的一个问题是**对离群点很敏感**，因为其对数概率随与中心的距离只按二次方式衰减。一个更稳健的替代物是 **Student-t 分布**（见第 ?? 节）。与高斯不同，即使在**没有缺失数据**的情况下，Student 的 MLE 也没有闭式表达式，因此我们必须求助于迭代优化方法。最容易使用的是 EM，因为它会自动保证 ν 为正，且 Σ 为对称正定。并且，由此得到的算法形式非常直观，如下所示。

乍看之下，也许不清楚为什么在**没有缺失数据**时还能用 EM。关键思想是**引入一个“人工”的隐藏/辅助变量来简化算法**。具体地，我们将利用这样一个事实：Student 分布可以写成**高斯尺度混合**（Gaussian scale mixture）：

$$T(x_i | \mu, \Sigma, \nu) = \int N(x_i | \mu, \Sigma/z_i) \text{Ga}(z_i | \frac{\nu}{2}, \frac{\nu}{2}) dz_i \quad (11.61)$$

(1 维情形的证明见练习 11.1。) 这可以看作是“无限多个”高斯的混合，每个都有稍微不同的协方差矩阵。把 z_i 当作缺失数据处理，**完备数据**的对数似然可写成

$$\ell_c(\theta) = \sum_{i=1}^N [\log N(x_i | \mu, \Sigma/z_i) + \log \text{Ga}(z_i | \nu/2, \nu/2)] \quad (11.62)$$

$$= \sum_{i=1}^N [-\frac{D}{2} \log(2\pi) - \frac{1}{2} \log |\Sigma| - \frac{z_i}{2} \delta_i + \frac{\nu}{2} \log \frac{\nu}{2} - \log \Gamma(\frac{\nu}{2})] \quad (11.63)$$

$$+ \frac{\nu}{2} (\log z_i - z_i) + (\frac{D}{2} - 1) \log z_i \quad (11.64)$$

其中我们把 **Mahalanobis 距离**定义为

$$\delta_i = (x_i - \mu)^\top \Sigma^{-1} (x_i - \mu). \quad (11.65)$$

我们可以把它分成两部分：一部分只含 μ 与 Σ ，另一部分只含 ν 。略去与优化无关的常数，有

$$\ell_c(\theta) = L_N(\mu, \Sigma) + L_G(\nu) \quad (11.66)$$

$$L_N(\mu, \Sigma) = -\frac{N}{2} \log |\Sigma| - \frac{1}{2} \sum_{i=1}^N z_i \delta_i \quad (11.67)$$

$$L_G(\nu) = -N \log \Gamma(\nu/2) + \frac{N\nu}{2} \log(\nu/2) + \frac{\nu}{2} \sum_{i=1}^N (\log z_i - z_i). \quad (11.68)$$

通俗解释

1) 为什么 t 分布“更稳健”?

高斯的尾巴很“轻”（概率衰减得快），离群点一旦远离中心， $(x - \mu)^2$ 很大，会对参数估计产生很大影响。t 分布尾巴更“重”（衰减慢），因此自然对离群点降权。

2) 怎么用 EM? “没有缺失”如何引入隐藏量?

把 t 分布看成：先从 Gamma 分布抽一个缩放因子 z_i ，再在协方差 Σ/z_i 的高斯中生成 x_i 。

- z_i 大 \Rightarrow 协方差小 \Rightarrow 该点“可信度高”；
 - z_i 小 \Rightarrow 协方差大 \Rightarrow 该点“可信度低”（更像离群点）。
- 这样就人为地造出了“隐藏变量” z_i ，可以套 EM。

3) E 步 (直观公式)

由 (11.61) 可得后验

$$z_i | x_i \sim \text{Ga}\left(\frac{\nu + D}{2}, \frac{\nu + \delta_i}{2}\right),$$

因此

$$\underbrace{w_i}_{:=\text{E}[z_i|x_i]} = \frac{\nu + D}{\nu + \delta_i}, \quad \underbrace{u_i}_{:=\text{E}[\log z_i|x_i]} = \psi\left(\frac{\nu + D}{2}\right) - \log\left(\frac{\nu + \delta_i}{2}\right),$$

其中 ψ 是 digamma。要点： δ_i 大（远离均值）的点 $\Rightarrow w_i$ 小 \Rightarrow 在下一步被自动降权。

4) M 步

- 由 L_N 可知这是“带权高斯 MLE”：

$$\mu = \frac{\sum_i w_i x_i}{\sum_i w_i}, \quad \Sigma = \frac{1}{N} \sum_i w_i (x_i - \mu)(x_i - \mu)^\top.$$

（也有把分母写成 $\sum_i w_i$ 的等价形式，差别仅在整体尺度；这里与 (11.67) 的写法一致。）

- 由 L_G 更新自由度 ν ：把 E 步得到的 u_i, w_i 代入 (11.68)，对 ν 求极大，即解标量方程

$$-\psi(\nu/2) + \log(\nu/2) + 1 + \frac{1}{N} \sum_i (u_i - w_i) = 0,$$

用 Newton/割线法数值求解即可（也可固定 ν 不更新，常见做法）。

5) 算法直觉与优点

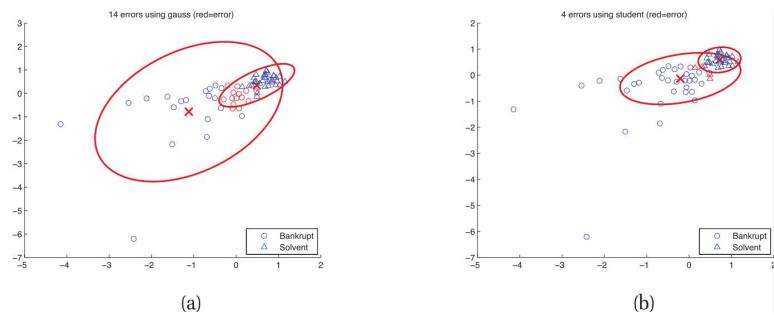
- 离群点的 δ_i 大 $\Rightarrow w_i$ 小 \Rightarrow 在均值/协方差更新时被弱化影响；
- EM 自动保证 $\nu > 0$ 、 $\Sigma > 0$ ，并给出简单的“加权高斯”更新；
- 当 $\nu \rightarrow \infty$ 时，t 分布 \rightarrow 高斯；当 ν 较小时，尾部更重，更稳健。

要点小结

- t 分布可写成 高斯 + Gamma 的尺度混合：式 (11.61)。
- 构造隐藏变量 z_i 后，完备对数似然分成两块：
 - $L_N(\mu, \Sigma)$: 像“带权高斯”——权重就是 z_i 。
 - $L_G(v)$: 只与 v 有关 (要用 $E[\log z_i], E[z_i]$)。
- **E 步**: $z_i \mid x_i \sim \text{Ga}((v + D)/2, (v + \delta_i)/2)$;
 $E[z_i] = (v + D)/(v + \delta_i)$ (远点被降权)。
- **M 步**:

$$\mu = \frac{\sum w_i x_i}{\sum w_i},$$

$$\Sigma = \frac{1}{N} \sum w_i (\cdot),$$
 v 解一维方程 (或固定)。
- **意义**: EM 给 t 分布提供了**稳健估计**: 自动下调离群点的作用，比纯高斯更不易被极端点“拖偏”。



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。

Figure 11.14 Mixture modeling on the bankruptcy data set. Left: Gaussian class conditional densities. Right: Student class conditional densities. Points that belong to class 1 are shown as triangles, points that belong to class 2 are shown as circles. The estimated labels, based on the posterior probability of belonging to each mixture component, are computed. If these are incorrect, the point is colored red, otherwise it is colored blue. (Training data is in black.)

11.4.5.1 EM with v known

Let us first derive the algorithm with v assumed known, for simplicity. In this case, we can ignore the LG term, so we only need to figure out how to compute $E[z_i]$ wrt the old parameters. From Section 4.6.2.2 we have

$$p(z_i|x_i, \theta) = Ga(z_i|v + D/2, v + \delta_i/2) \quad (11.69)$$

Now if $z_i \sim Ga(a, b)$, then $E[z_i] = a/b$. Hence the E step at iteration t is

$$z(t)_i = E[z_i|x_i, \theta(t)] = v(t) + D/v(t) + \delta(t)/i \quad (11.70)$$

The M step is obtained by maximizing $E[\ln(\mu, \Sigma)]$ to yield

$$\hat{\mu}^{(t+1)} = \frac{1}{N} \sum z(t)_i x_i / z(t)_i \quad (11.71)$$

$$\hat{\Sigma}^{(t+1)} = \frac{1}{N} \sum z(t)_i x_i x_i^T - \frac{1}{N} \sum z(t)_i \hat{\mu}^{(t+1)} (\hat{\mu}^{(t+1)})^T \quad (11.72)$$

$$= \frac{1}{N} \sum z(t)_i x_i x_i^T - \frac{1}{N} \sum i z(t)_i \hat{\mu}^{(t+1)} (\hat{\mu}^{(t+1)})^T \quad (11.73)$$

These results are quite intuitive: the quantity z_i is the precision of measurement i , so if it is small, the corresponding data point is down-weighted when estimating the mean and covariance. This is how the Student achieves robustness to outliers.

11.4.5.2 EM with v unknown

To compute the MLE for the degrees of freedom, we first need to compute the expectation of $LG(v)$, which involves z_i and $\log z_i$. Now if $z_i \sim Ga(a, b)$, then one can show that

$$(t)_i = E[\log z_i|\theta(t)] = \Psi(a) - \log b \quad (11.74)$$

where $\Psi(x) = d/dx \log \Gamma(x)$ is the digamma function. Hence, from Equation 11.69, we have

$$(t)_i = \Psi(v(t) + D/2) - \log(v(t) + \delta(t)/i/2) \quad (11.75)$$

$$= \log(z(t)_i) + \Psi(v(t) + D/2) - \log(v(t) + D/2) \quad (11.76)$$

Substituting into Equation 11.68, we have

$$E[LG(v)] = -N \log \Gamma(v/2) + Nv/2 \log(v/2) + v/2 \sum (t)_i - z(t)_i \quad (11.77)$$

The gradient of this expression is equal to

$$\frac{d}{dv} E[LG(v)] = -N/2 \Psi(v/2) + N/2 \log(v/2) + N/2 + 1/2 \sum ((t)_i - z(t)_i) \quad (11.78)$$

This has a unique solution in the interval $(0, +\infty]$ which can be found using a 1d constrained optimizer.

Performing a gradient-based optimization in the M step, rather than a closed-form update, is an example of what is known as the generalized EM algorithm. One can show that EM will still converge to a local optimum even if we only perform a “partial” improvement to the parameters in the M step.

11.4.5.3 Mixtures of Student distributions

It is easy to extend the above methods to fit a mixture of Student distributions. See Exercise 11.4 for the details. Let

us consider a small example from (Lo 2009, ch3). We have a N = 66, D = 2 data set regarding the bankruptcy patterns of certain companies. The first feature specifies the ratio of retained earnings (RE) to total assets, and the second feature specifies the ratio of earnings before interests and taxes (EBIT) to total assets. We fit two models to this data, ignoring the class labels: a mixture of 2 Gaussians, and a mixture of 2 Students. We then use each fitted model to classify the data. We compute the most probable cluster membership and treat this as \hat{y}_i . We then compare \hat{y}_i to the true labels y_i and compute an error rate. If this is more than 50%, we permute the latent labels (i.e., we consider cluster 1 to represent class 2 and vice versa), and then recompute the error rate. Points which are misclassified are then shown in red. The result is shown in Figure 11.14. We see that the Student model made 4 errors, the Gaussian model made 21. This is because the class-conditional densities contain some extreme values, causing the Gaussian to be a poor choice.

下面先把原文逐句完整翻译，然后用通俗方式解释，最后给出要点小结。

翻译

图 11.14 在破产数据集上的混合建模。左：高斯类条件分布；右：Student 类条件分布。属于类别 1 的点用三角形表示，属于类别 2 的点用圆圈表示。我们根据属于各个混合分量的后验概率来计算估计标签；若估计错误，该点涂成红色，否则涂成蓝色。（训练数据用黑色表示。）

11.4.5.1 已知 v 的 EM

为简化起见，先推导把自由度 v 视为已知时的算法。在这种情况下，我们可以忽略 L_G 项，所以只需要用旧参数来计算 $E[z_i]$ 。由 4.6.2.2 节可得

$$p(z_i | x_i, \theta) = \text{Ga}(z_i | \frac{v+D}{2}, \frac{v+\delta_i}{2}) \quad (11.69)$$

若 $z_i \sim \text{Ga}(a, b)$ ，则 $E[z_i] = a/b$ 。因此在第 t 次迭代的 E 步为

$$z_i^{(t)} = E[z_i \mid x_i, \theta^{(t)}] = \frac{\nu^{(t)} + D}{\nu^{(t)} + \delta_i^{(t)}}. \quad (11.70)$$

M 步通过最大化 $E[L_N(\mu, \Sigma)]$ 得到

$$\hat{\mu}^{(t+1)} = \frac{\sum_i z_i^{(t)} x_i}{\sum_i z_i^{(t)}} \quad (11.71)$$

$$\hat{\Sigma}^{(t+1)} = \frac{1}{N} \sum_i z_i^{(t)} (x_i - \hat{\mu}^{(t+1)}) (x_i - \hat{\mu}^{(t+1)})^\top = \frac{1}{N} \sum_i z_i^{(t)} x_i x_i^\top - \frac{1}{N} (\sum_{i=1}^N z_i^{(t)}) \hat{\mu}^{(t+1)} \hat{\mu}^{(t+1)^\top}$$

这些结果具有直观含义： z_i 是第 i 个观测的“精度”，如果它很小，那么在估计均值和协方差时，该数据点就会被降低权重。Student 分布正是通过这种方式对离群点具有稳健性。

11.4.5.2 未知 ν 的 EM

要计算自由度的极大似然估计，首先需要计算 $L_G(\nu)$ 的期望，其中涉及 z_i 与 $\log z_i$ 。若 $z_i \sim \text{Ga}(a, b)$ ，可得

$$\eta_i^{(t)} := E[\log z_i \mid \theta^{(t)}] = \Psi(a) - \log b, \quad (11.74)$$

其中 $\Psi(x) = \frac{d}{dx} \log \Gamma(x)$ 是 digamma 函数。因此由式 (11.69)，

$$\eta_i^{(t)} = \Psi\left(\frac{\nu^{(t)} + D}{2}\right) - \log\left(\frac{\nu^{(t)} + \delta_i^{(t)}}{2}\right) \quad (11.75)$$

$$= \log z_i^{(t)} + \Psi\left(\frac{\nu^{(t)} + D}{2}\right) - \log\left(\frac{\nu^{(t)} + D}{2}\right). \quad (11.76)$$

代入式 (11.68)，得到

$$E[L_G(\nu)] = -N \log \Gamma(\nu/2) + \frac{N\nu}{2} \log(\nu/2) + \frac{\nu}{2} \sum_i (\eta_i^{(t)} - z_i^{(t)}). \quad (11.77)$$

其关于 ν 的梯度为

$$\frac{d}{d\nu} E[L_G(\nu)] = -\frac{N}{2} \Psi(\nu/2) + \frac{N}{2} \log(\nu/2) + \frac{N}{2} + \frac{1}{2} \sum_i (\eta_i^{(t)} - z_i^{(t)}). \quad (11.78)$$

在区间 $(0, +\infty]$ 上该方程有唯一解，可用一维有约束优化器求解。在 M 步中使用基于梯度的数值优化（而不是闭式更新），属于所谓广义 EM (GEM)。即使 M 步只对参数做“部分改进”，也能证明 EM 仍会收敛到一个局部极值。

11.4.5.3 Student 混合模型

把上述方法推广到Student 混合很容易；细节见练习 11.4。看一个来自 (Lo 2009, 第 3 章) 的小示例：有一个 $N = 66, D = 2$ 的数据集，描述一些公司的破产模式。第一个特征是**留存收益/总资产**，第二个特征是**息税前利润 (EBIT) /总资产**。我们在忽略类别标签的情况下，对该数据分别拟合两个模型：**2 个高斯的混合与2 个 Student 的混合**。然后用各自的拟合模型对数据进行分类：计算每个点最可能的簇归属，把它当作 \hat{y}_i 。将 \hat{y}_i 与真实标签 y_i 比较得到错误率；若错误率超过 50%，我们对潜在簇的标签进行置换（即把簇 1 当作类别 2，反之亦然），再重新计算错误率。把被误分的点画成红色。结果如图 11.14：Student 模型错 4 个点，高斯模型错 21 个点。原因在于类条件分布中包含一些**极端值**，这使得高斯成为一个不佳的选择。

通俗解释

- **为什么 Student 比高斯“抗离群点”？**

在 EM 的 E 步里，我们给每个样本算一个隐变量 z_i （精度/权重）。离均值远的样本 δ_i 大 $\Rightarrow z_i$ 小 \Rightarrow 在 M 步里更新 μ, Σ 时权重变小，所以离群点不会把参数“拉歪”。

- **已知 ν 的时候算法特别简单**

就是“**加权高斯 MLE**”：

均值 = 权重加权的平均；协方差 = 加权的离差矩阵。权重 $z_i = (\nu + D)/(\nu + \delta_i)$ 。

- **未知 ν 怎么办？**

用 GEM：把 $E[\log z_i]$ 也算出来（用 digamma），把它们放进只依赖 ν 的那一项里，对 ν 做一维数值优化即可（每步不必找到精确极大，只要向好方向走）。

- **实际效果（图 11.14）**

这个破产数据有不少“极端”公司（远离主要团簇）。高斯很容易被这些点影响，从而划错很多；Student 自动给这些点低权重，错误数明显更少。

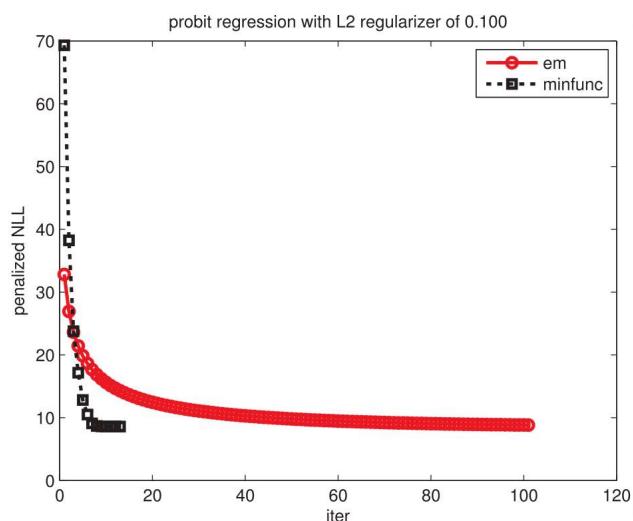
要点小结

- Student 分布可写成“**高斯协方差被 Gamma 随机缩放**的尺度混合”；可引入隐变量 z_i 用 EM。
- **E 步**： $z_i \mid x_i \sim \text{Ga}((\nu + D)/2, (\nu + \delta_i)/2)$ ，权重 $E[z_i] = (\nu + D)/(\nu + \delta_i)$ （离群点被降权）。
- **M 步 (ν 已知)**：

$$\mu = \frac{\sum z_i x_i}{\sum z_i},$$

$$\Sigma = \frac{1}{N} \sum z_i (x_i - \mu)(x_i - \mu)^T.$$

- **M步 (v未知)**：用 $\eta_i = E[\log z_i]$ 与 z_i 代入 $E[L_G(v)]$, 对 v 做一维优化 (GEM)。
- **Student 混合**：按上法对每个分量做 E/M；在含极端值的数据上通常优于高斯混合（示例中错 4 vs 21）。
- 直观记忆：t-EM = 加权高斯 EM，权重由“离中心的远近”自动决定，从而实现对离群点的稳健性。



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。

Figure 11.15 Fitting a probit regression model in 2d using a quasi-Newton method or EM. Figure generated by probitRegDemo.

11.4.6 EM for probit regression

In Section 9.4.2, we described the latent variable interpretation of probit regression. Recall that this has the form $p(y_i = 1|z_i) = I(z_i > 0)$, where $z_i \sim N(w^T x_i, 1)$ is latent. We now show how to fit this model using EM. (Although it is possible to fit probit regression models using gradient based methods, as shown in Section 9.4.1, this EM-based approach has the advantage that it generalized to many other kinds of models, as we will see later on.) The complete data log likelihood has the following form, assuming a $N(0, V_0)$ prior on w :

$$(z, w | V_0) = \log p(y|z) + \log N(z|Xw, I) + \log N(w|0, V_0) \quad (11.79)$$

$$= i \log p(y_i|z_i) - \frac{1}{2} (z_i - Xw)^T (z_i - Xw) - \frac{1}{2} w^T V^{-1} w + \text{const} \quad (11.80)$$

The posterior in the E step is a truncated Gaussian:

$$p(z_i|y_i, x_i, w) = N(z_i|w^T x_i, 1)I(z_i > 0) \text{ if } y_i = 1 \\ N(z_i|w^T x_i, 1)I(z_i < 0) \text{ if } y_i = 0 \quad (11.81)$$

In Equation 11.80, we see that w only depends linearly on z , so we just need to compute $E[z_i|y_i, x_i, w]$. Exercise 11.15 asks you to show that the posterior mean is given by

$$E[z_i|w, x_i] = \mu_i + \varphi(\mu_i) 1 - \Phi(-\mu_i) = \mu_i + \varphi(\mu_i) \Phi(\mu_i) \text{ if } y_i = 1 \\ \mu_i - \varphi(\mu_i) \Phi(-\mu_i) = \mu_i - \varphi(\mu_i) 1 - \Phi(\mu_i) \text{ if } y_i = 0 \quad (11.82)$$

where $\mu_i = w^T x_i$. In the M step, we estimate w using ridge regression, where $\mu = E[z]$ is the output we are trying to predict. Specifically, we have

$$\hat{w} = (V^{-1} I + X^T X)^{-1} X^T \mu \quad (11.83)$$

The EM algorithm is simple, but can be much slower than direct gradient methods, as illustrated in Figure 11.15. This is because the posterior entropy in the E step is quite high, since we only observe that z is positive or negative, but are given no information from the likelihood about its magnitude. Using a stronger regularizer can help speed convergence, because it constrains the range of plausible z values. In addition, one can use various speedup tricks, such as data augmentation (van Dyk and Meng 2001), but we do not discuss that here.

下面先把原文完整翻译（不遗漏），再用通俗方式解释，最后给出要点小结。

翻译

图 11.15 用拟牛顿法或 EM 在二维上拟合一个 probit 回归模型。图由 `probitRegDemo` 生成。

11.4.6 用 EM 做 probit 回归

在第 9.4.2 节里，我们给出了 probit 回归的**潜变量解释**。回忆其形式为

$$p(y_i = 1 | z_i) = \mathbf{1}(z_i > 0), \text{ 其中潜变量 } z_i \sim N(w^T x_i, 1).$$

现在我们说明如何用 EM 拟合这个模型。（虽然按第 9.4.1 节所示，可以用基于梯度的方法来拟合 probit 回归，但这种基于 EM 的做法的优点是：它能推广到许多其他模型，后文会看到。）

设对 w 的先验为 $N(0, V_0)$ 。**完备数据对数似然**为

$$\ell(z, w \mid V_0) = \log p(y \mid z) + \log N(z \mid Xw, I) + \log N(w \mid 0, V_0) \quad (11.79)$$

$$= \sum_i \log p(y_i \mid z_i) - \frac{1}{2}(z - Xw)^\top (z - Xw) - \frac{1}{2}w^\top V_0^{-1}w + \text{const.} \quad (11.80)$$

E 步中的后验是截断高斯：

$$p(z_i \mid y_i, x_i, w) = \begin{cases} N(z_i \mid w^\top x_i, 1) \mathbf{1}(z_i > 0), & y_i = 1 \\ N(z_i \mid w^\top x_i, 1) \mathbf{1}(z_i < 0), & y_i = 0 \end{cases} \quad (11.81)$$

在式 (11.80) 中可以看到， w 只线性地依赖于 z ，因此我们只需要计算 $E[z_i \mid y_i, x_i, w]$ 。

练习 11.15 要你证明其后验均值为

$$E[z_i \mid w, x_i] = \begin{cases} \mu_i + \frac{\phi(\mu_i)}{1 - \Phi(-\mu_i)} = \mu_i + \frac{\phi(\mu_i)}{\Phi(\mu_i)}, & y_i = 1 \\ \mu_i - \frac{\phi(\mu_i)}{\Phi(-\mu_i)} = \mu_i - \frac{\phi(\mu_i)}{1 - \Phi(\mu_i)}, & y_i = 0 \end{cases} \quad (11.82)$$

其中 $\mu_i = w^\top x_i$, ϕ 与 Φ 分别为标准正态的密度与分布函数。

在 M 步里，我们用岭回归来估计 w ，其中要预测的输出是 $\mu = E[z]$ 。具体为

$$\hat{w} = (V_0^{-1} + X^\top X)^{-1} X^\top \mu. \quad (11.83)$$

EM 算法很简单，但相比直接的梯度法可能慢得多，如图 11.15 所示。这是因为在 E 步里后验熵很大：我们只观察到 z 的符号是正还是负，而似然对它的大小并没有提供信息。更强的正则化可以帮助加速收敛，因为它限制了 z 的可能范围。另外也可以用一些加速技巧，比如数据增广 (van Dyk 和 Meng, 2001)，但这里不展开。

通俗解释

1) probit 的潜变量图景

- 设有一个隐藏的连续变量 z_i 。当 $z_i > 0$ 时 $y_i = 1$ ，否则 $y_i = 0$ 。
- z_i 围绕 $w^\top x_i$ 做标准差为 1 的高斯波动。

2) 为什么可以用 EM?

- 若我们“看见”了 z ，就是一个带高斯噪声的线性回归，有闭式解。
- 实际看不见，于是在 E 步用当前 w 计算每个样本的截断高斯后验均值 $E[z_i \mid y_i, x_i, w]$ ；
- 在 M 步把这些 $E[z_i]$ 当作回归的“目标值”，做一次岭回归得新 w 。循环直到收敛。

3) 式 (11.82) 的直觉

- 当 $y_i = 1$ 时, 我们知道 z_i 必为正, 于是把原本以 $\mu_i = w^\top x_i$ 为中心的高斯截断到 > 0 ; 其均值在 μ_i 的基础上往右被拉了一点, 拉动大小是 $\phi(\mu_i)/\Phi(\mu_i)$ 。
- 当 $y_i = 0$ 时, 均值往左被拉: $\mu_i - \phi(\mu_i)/(1 - \Phi(\mu_i))$ 。

4) 为何 EM 往往比梯度法慢?

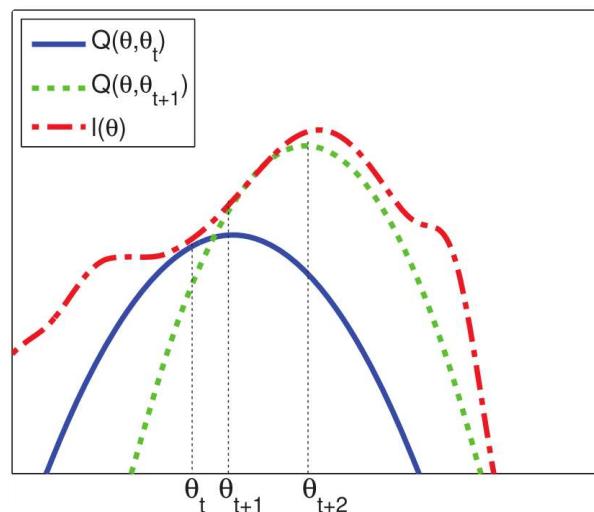
- 我们对 z 只知道正/负, 幅度不知 \Rightarrow 不确定性大 \Rightarrow 每次 E 步给的“软目标”信息不足, 收敛慢。
- 加强 L_2 正则 (减小 V_0) 能压缩 z 的范围, 加快迭代; 也可用数据增广等技巧再提速。

5) 图 11.15 怎么读?

- 横轴迭代次数, 纵轴是带正则的 NLL。
- 黑色虚线 (拟牛顿) 几步就下到较低值; 红线 (EM) 单调下降但需要很多轮才接近。

要点小结

- **模型:** $z_i \sim N(w^\top x_i, 1)$, $y_i = \mathbf{1}(z_i > 0)$, 先验 $w \sim N(0, V_0)$ 。
- **E 步:** 对每个样本求截断高斯的均值 $E[z_i]$, 公式见 (11.82)。
- **M 步:** 把 $\mu = E[z]$ 当回归目标, 做岭回归更新 w : $\hat{w} = (V_0^{-1} + X^\top X)^{-1} X^\top \mu$ 。
- **性质:** EM 简单、单调下降但可能慢; 正则更强收敛更快; 也可用数据增广等技巧提速。
- **比较:** 对 probit 回归, 梯度/拟牛顿通常更快; EM 的优势在于可推广到更复杂的潜变量模型。



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。

Figure 11.16 Illustration of EM as a bound optimization algorithm. Based on Figure 9.14 of (Bishop 2006a). Figure generated by emLogLikelihoodMax.

11.4.7 Theoretical basis for EM

In this section, we show that EM monotonically increases the observed data log likelihood until it reaches a local maximum (or saddle point, although such points are usually unstable). Our derivation will also serve as the basis for various generalizations of EM that we will discuss later.

11.4.7.1 Expected complete data log likelihood is a lower bound

Consider an arbitrary distribution $q(z_i)$ over the hidden variables. The observed data log likelihood can be written as follows:

$$l(\theta) = \sum_{i=1}^N \log p(x_i, z_i | \theta) = \sum_{i=1}^N \log q(z_i) p(x_i, z_i | \theta) \quad (11.84)$$

Now $\log(u)$ is a concave function, so from Jensen's inequality (Equation 2.113) we have the following lower bound:

$$(11.85) \quad l(\theta) \geq \sum_{i=1}^N q_i(z_i) \log p(x_i, z_i | \theta)$$

Let us denote this lower bound as follows:

$$Q(\theta, q) = \sum_{i=1}^N q_i(z_i) \log p(x_i, z_i | \theta) + H(q_i) \quad (11.86)$$

where $H(q_i)$ is the entropy of q_i . The above argument holds for any positive distribution q . Which one should we choose? Intuitively we should pick the q that yields the tightest lower bound. The lower bound is a sum over i of terms of the following form:

$$L(\theta, q_i) = q_i(z_i) \log p(x_i, z_i | \theta) \quad (11.87)$$

$$= q_i(z_i) \log p(z_i | x_i, \theta) p(x_i | \theta) \quad (11.88)$$

$$= q_i(z_i) \log p(z_i | x_i, \theta) + q_i(z_i) \log p(x_i | \theta) \quad (11.89)$$

$$= -KL(q_i(z_i) || p(z_i | x_i, \theta)) + \log p(x_i | \theta) \quad (11.90)$$

The $p(x_i | \theta)$ term is independent of q_i , so we can maximize the lower bound by setting $q_i(z_i) = p(z_i | x_i, \theta)$. Of course, θ is unknown, so instead we use $q_t(z_i) = p(z_i | x_i, \theta_t)$, where θ_t is our estimate of the parameters at iteration t . This is the output of the E step. Plugging this in to the lower bound we get

$$Q(\theta, q_t) = \sum_{i=1}^N q_t(z_i) \log p(x_i, z_i | \theta) + H(q_t) \quad (11.91)$$

We recognize the first term as the expected complete data log likelihood. The second term is a constant wrt θ . So the M step becomes

$$\theta_{t+1} = \operatorname{argmax}_{\theta} Q(\theta, \theta_t) = \arg \max_{\theta} \sum_i E_{q(\cdot|z_i)} [\log p(x_i, z_i|\theta)] \quad (11.92)$$

as usual. Now comes the punchline. Since we used $q(z_i) = p(z_i|x_i, \theta_t)$, the KL divergence becomes zero, so $L(\theta_t, q_i) = \log p(x_i|\theta_t)$, and hence

$$Q(\theta_t, \theta_t) = \sum_i \log p(x_i|\theta_t) = L(\theta_t) \quad (11.93)$$

We see that the lower bound is tight after the E step.

Since the lower bound “touches” the function, maximizing the lower bound will also “push up” on the function itself. That is, the M step is guaranteed to modify the parameters so as to increase the likelihood of the observed data (unless it is already at a local maximum).

This process is sketched in Figure 11.16. The dashed red curve is the original function (the observed data log-likelihood). The solid blue curve is the lower bound, evaluated at θ_t ; this touches the objective function at θ_t . We then set θ_{t+1} to the maximum of the lower bound (blue curve), and fit a new bound at that point (dotted green curve). The maximum of this new bound becomes θ_{t+2} , etc. (Compare this to Newton’s method in Figure 8.4(a), which repeatedly fits and then optimizes a quadratic approximation.)

11.4.7.2 EM monotonically increases the observed data log likelihood

We now prove that EM monotonically increases the observed data log likelihood until it reaches a local optimum. We have

$$Q(\theta_{t+1}) \geq Q(\theta_t, \theta_t) \geq Q(\theta_t, \theta_t) = L(\theta_t) \quad (11.94)$$

where the first inequality follows since $Q(\theta, \cdot)$ is a lower bound on (θ) ; the second inequality follows since, by definition, $Q(\theta_{t+1}, \theta_t) = \max_{\theta} Q(\theta, \theta_t) \geq Q(\theta_t, \theta_t)$; and the final equality follows Equation 11.93.

As a consequence of this result, if you do not observe monotonic increase of the observed data log likelihood, you must have an error in your math and/or code. (If you are performing MAP estimation, you must add on the log prior term to the objective.) This is a surprisingly powerful debugging tool.

下面先把原文逐句不遗漏地译成中文，然后用通俗解释，最后给出要点小结。

翻译

图 11.16 把 EM 看成一个“下界优化”算法的示意图。基于 (Bishop 2006a) 的图 9.14。图由 `emLogLikelihoodMax` 生成。

11.4.7 EM 的理论基础

本节我们证明：EM 会单调地提高观测数据的对数似然，直达到一个局部极大值（或鞍点，但这种点通常不稳定）。我们的推导也将作为后面若干 EM 推广方法的基础。

11.4.7.1 期望完备数据对数似然是一个下界

考虑一个关于隐藏变量的任意分布 $q(z_i)$ 。观测数据的对数似然可以写成

$$\ell(\theta) = \sum_{i=1}^N \log \sum_{z_i} p(x_i, z_i | \theta) = \sum_{i=1}^N \log \sum_{z_i} q(z_i) \frac{p(x_i, z_i | \theta)}{q(z_i)} \quad (11.84)$$

由于 $\log(u)$ 是凹函数，由 Jensen 不等式（式 2.113）得到如下下界：

$$\ell(\theta) \geq \sum_i \sum_{z_i} q_i(z_i) \log \frac{p(x_i, z_i | \theta)}{q_i(z_i)} \quad (11.85)$$

把这个下界记作

$$Q(\theta, q) = \sum_i E_{q_i} [\log p(x_i, z_i | \theta)] + H(q_i) \quad (11.86)$$

其中 $H(q_i)$ 是 q_i 的熵。以上论证对任何正的分布 q 都成立。那我们该选哪一个 q 呢？直觉上应选择使下界最紧的 q 。下界是若干项之和，每项形如

$$L(\theta, q_i) = \sum_{z_i} q_i(z_i) \log \frac{p(x_i, z_i | \theta)}{q_i(z_i)} \quad (11.87)$$

$$= \sum_{z_i} q_i(z_i) \log \frac{p(z_i | x_i, \theta) p(x_i | \theta)}{q_i(z_i)} \quad (11.88)$$

$$= \sum_{z_i} q_i(z_i) \log \frac{p(z_i | x_i, \theta)}{q_i(z_i)} + \sum_{z_i} q_i(z_i) \log p(x_i | \theta) \quad (11.89)$$

$$= -KL(q_i(z_i) \| p(z_i | x_i, \theta)) + \log p(x_i | \theta). \quad (11.90)$$

其中 $\log p(x_i | \theta)$ 与 q_i 无关，因此要最大化下界，只需令

$q_i(z_i) = p(z_i | x_i, \theta)$ 。当然 θ 未知，于是改用

$q_i^t(z_i) = p(z_i | x_i, \theta_t)$, 其中 θ_t 是第 t 次迭代时的参数估计。这正是 E 步的输出。把它代回下界得到

$$Q(\theta, q^t) = \sum_i \mathbb{E}_{q_i^t} [\log p(x_i, z_i | \theta)] + H(q_i^t). \quad (11.91)$$

我们认出第一项就是期望完备数据对数似然；第二项与 θ 无关。因此 M 步变成

$$\theta_{t+1} = \arg \max_{\theta} Q(\theta, \theta_t) = \arg \max_{\theta} \sum_i \mathbb{E}_{q_i^t} [\log p(x_i, z_i | \theta)] \quad (11.92)$$

——与通常写法一致。关键来了：因为我们取的是

$q_i^t(z_i) = p(z_i | x_i, \theta_t)$, 所以 KL 散度为 0, 于是

$L(\theta_t, q_i) = \log p(x_i | \theta_t)$, 从而

$$Q(\theta_t, \theta_t) = \sum_i \log p(x_i | \theta_t) = \ell(\theta_t). \quad (11.93)$$

可见在 E 步之后，这个下界与原目标函数在 θ_t 处是贴合 (tight) 的。既然下界“贴住”了目标函数，最大化下界也会把目标函数“往上顶”。也就是说，除非已经到达局部极大点，M 步一定会把观测数据的对数似然提高。

图 11.16 对此作了草图说明：红色虚线是原函数（观测数据对数似然）；蓝色实线是以 θ_t 为基点的下界，在 θ_t 处与目标函数相切。把下界最大化得到 θ_{t+1} ，再在该点拟合新的下界（绿色虚线），其最大值为 θ_{t+2} ，如此往复。（可与图 8.4(a) 的牛顿法对比：后者反复拟合并优化二次近似。）

11.4.7.2 EM 单调提高观测数据的对数似然

下面证明 EM 会单调提高观测数据的对数似然直至到达局部最优。我们有

$$\ell(\theta_{t+1}) \geq Q(\theta_{t+1}, \theta_t) \geq Q(\theta_t, \theta_t) = \ell(\theta_t) \quad (11.94)$$

其中第一步不等式来自： $Q(\theta, \cdot)$ 是 $\ell(\theta)$ 的下界；第二步来自定义： $Q(\theta_{t+1}, \theta_t) = \max_{\theta} Q(\theta, \theta_t) \geq Q(\theta_t, \theta_t)$ ；最后的等式由式 (11.93) 得到。

因此，如果你在实现中没有看到观测数据对数似然单调上升，那就说明你的推导和/或代码里有错误。（做 MAP 时要把对数先验加到目标里。）这其实是一个非常有用的调试工具。

通俗解释

- EM 每一轮做两件事：
 - 1) E 步：用当前参数算出“最好的”隐藏变量分布 $q = p(z | x, \theta)$ 。这一步使得下界与

真目标在当前参数处完全贴合。

2) **M 步**: 在这个贴合的下界上做最大化, 得到新参数。因为下界 \leq 目标且在当前点相等, 所以下界的上升必然让目标也上升。

- 你可以把 EM 想像成: 不断给目标函数找一个**可优化的替身 (下界)**, 每次把替身推到最高, 再换一个在新位置贴合的替身继续推。图中蓝线/绿虚线就是两轮替身, 红虚线是真目标。
- 这与牛顿法“二次近似 + 最优化”的精神类似, 但 EM 用的是**凹函数** + Jensen得到的**可证明单调的下界**; 因此每步都不会把目标降下来 (除非数值或实现有误) 。

要点小结

- 通过 Jensen 不等式得到:

$$\ell(\theta) \geq Q(\theta, q) = \sum_i E_q[\log p(x_i, z_i | \theta)] + H(q).$$
- 取 $q = p(z | x, \theta_t)$ 让下界在 θ_t 处变紧:

$$Q(\theta_t, \theta_t) = \ell(\theta_t).$$
- E 步**: 计算 $q_i^t(z_i) = p(z_i | x_i, \theta_t)$ 。
- M 步**: 最大化 $Q(\theta, \theta_t) = \sum_i E_{q_i^t}[\log p(x_i, z_i | \theta)]$ 。
- 单调性**: $\ell(\theta_{t+1}) \geq \ell(\theta_t)$, 直到局部最优/鞍点。
- 实战提示: 若对数似然 (或 MAP 时的“对数似然 + 对数先验”) 没有单调上升, 基本可以判定是**推导或代码错误**。

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。

11.4.8 Online EM

When dealing with large or streaming datasets, it is important to be able to learn online, as we discussed in Section 8.5. There are two main approaches to online EM in the literature. The first approach, known as incremental EM (Neal and Hinton 1998), optimizes the lower bound $Q(\theta, q_1, \dots, q_N)$ one q_i at a time; however, this requires storing the expected sufficient statistics for each data case. The second approach, known as stepwise EM (Sato and Ishii 2000; Cappe and Mouline 2009; Cappe 2010), is based on stochastic approximation theory, and only

requires constant memory use. We explain both approaches in more detail below, following the presentation of (Liang and Klein Liang and Klein).

11.4.8.1 Batch EM review

Before explaining online EM, we review batch EM in a more abstract setting. Let $\varphi(x, z)$ be a vector of sufficient statistics for a single data case. (For example, for a mixture of multinoullis, this would be the count vector $a(j)$, which is the number of cluster j was used in z , plus the matrix $B(j, v)$, which is of the number of times the hidden state was j and the observed letter was v .) Let $s_i = z p(z|xi, \Theta) \varphi(xi, z)$ be the expected sufficient statistics for case i , and $\mu = N \sum_{i=1}^N s_i$ be the sum of the ESS. Given μ , we can derive an ML or MAP estimate of the parameters in the M step; we will denote this operation by $\theta(\mu)$. (For example, in the case of mixtures of multinoullis, we just need to normalize a and each row of B .) With this notation under our belt.

11.4.8.2 Incremental EM

In incremental EM (Neal and Hinton 1998), we keep track of μ as well as the s_i . When we come to a data case, we swap out the old s_i and replace it with the new s_{new} .

Note that we can exploit the sparsity of s_{new} to speedup the computation of θ , since most components of μ will not have changed.

This can be viewed as maximizing the lower bound $Q(\theta, q_1, \dots, q_N)$ by optimizing q_1 , then θ , then q_2 , then θ , etc. As such, this method is guaranteed to monotonically converge to a local maximum of the lower bound and to the log likelihood itself.

11.4.8.3 Stepwise EM

In stepwise EM, whenever we compute a new s_i , we move μ towards it. At iteration k , the stepsize has value η_k , which must satisfy the Robbins-Monro conditions in Equation 8.82. For example, (Liang and Klein Liang and Klein) use $\eta_k = (2 + k)^{-\kappa}$ for $0.5 < \kappa \leq 1$. We can get somewhat better behavior by using a minibatch of size m before each update. It is possible to optimize m and κ to maximize the training set likelihood, by trying different values in parallel for an initial trial period; this can significantly speed up the algorithm.

(Liang and Klein Liang and Klein) compare batch EM, incremental EM, and stepwise EM on four different unsupervised language modeling tasks. They found that

stepwise EM (using $\kappa \approx 0.7$ and $m \approx 1000$) was faster than incremental EM, and both were much faster than batch EM. In terms of accuracy, stepwise EM was usually as good or sometimes even better than batch EM; incremental EM was often worse than either of the other methods.

下面先把原文“逐字不遗漏”地译成中文；随后给出通俗解释与操作要点小结。

翻译

11.4.8 在线 EM (Online EM)

在处理**特别大或流式**的数据集时，正如第 8.5 节讨论的，能够**在线学习**很重要。文献中关于在线 EM 有两条主要路线。第一条叫**增量 EM** (incremental EM; Neal and Hinton, 1998)，它一次只对一个 q_i 来优化下界 $Q(\theta, q_1, \dots, q_N)$ ；但是这需要为每条数据样本**保存期望充分统计量**。第二条叫**逐步 EM** (stepwise EM; Sato and Ishii, 2000; Cappé and Moulines, 2009; Cappé, 2010)，它基于**随机逼近理论**，只需要**常数级内存**。下面我们按 (Liang and Klein) 的讲法更详细地解释这两种方法。

11.4.8.1 批量 (batch) EM 回顾

在介绍在线 EM 之前，先在一个更抽象的设置下回顾批量 EM。令 $\phi(x, z)$ 是单个样本的**充分统计量向量**。（例如，对于多项伯努利的混合模型， ϕ 包含计数向量 $a(j)$: 潜变量 z 中簇 j 被使用的次数；以及矩阵 $B(j, v)$: 隐藏状态为 j 且观测字母为 v 的次数。）

令

$$s_i = \sum_z p(z | x_i, \theta) \phi(x_i, z)$$

为第 i 个样本的**期望充分统计量 (ESS)**，并令

$$\mu = \sum_{i=1}^N s_i$$

为 ESS 的**总和**。给定 μ ，我们可以在 M 步里推出参数的 ML 或 MAP 估计；把这个操作记为 $\theta(\mu)$ 。（例如，对于多项伯努利的混合，只需要把 a 和 B 的每一行做一次**归一化**。）

11.4.8.2 增量 EM

在增量 EM (Neal and Hinton, 1998) 中，我们同时维护 μ 以及所有的 s_i 。当处理某个样本时，用新算出的 s_i^{new} 去替换旧的 s_i ，并相应更新 μ 。注意可利用 s_i^{new} 的**稀疏性**加速 θ

的计算，因为 μ 的大多数分量并不会改变。

这可以看作是通过依次优化 q_1 、然后 θ 、再 q_2 、再 θ 、…… 来最大化下界

$Q(\theta, q_1, \dots, q_N)$ 。因此，这个方法被保证能**单调地收敛**到下界的一个局部极大值，也即到对数似然本身的局部极大值。

11.4.8.3 逐步 EM

在逐步 EM 中，每当我们算出一个新的 s_i 时，就把 μ 向它移动。在第 k 次迭代，步长为 η_k ，它必须满足 Robbins–Monro 条件（式 8.82）。例如，(Liang and Klein) 采用

$$\eta_k = (2 + k)^{-\kappa}, \quad 0.5 < \kappa \leq 1.$$

在每次更新前使用大小为 m 的**小批** (minibatch) 能得到更好的表现。可以在初始的试验阶段并行尝试不同的 m 与 κ 来最大化训练集似然；这能显著加速算法。

(Liang and Klein) 在四个无监督语言建模任务上比较了**批量 EM**、**增量 EM**、**逐步 EM**。他们发现：逐步 EM ($\kappa \approx 0.7$, $m \approx 1000$) 比增量 EM **更快**，而这两者都**远快于**批量 EM。就精度而言，逐步 EM 通常与批量 EM 一样好，有时甚至**更好**；增量 EM 往往不如另外两种方法。

通俗解释

- **核心对象**：

- s_i : 第 i 个样本的“软计数”（期望充分统计量）；
- $\mu = \sum_i s_i$: 把所有样本的软计数加在一起；
- $\theta(\mu)$: 给定总软计数 μ 后，按 M 步的**闭式更新/正规化**得到的参数。

- **批量 EM**: 先用所有数据算完所有 s_i ，汇总成 μ ，再一次性更新 θ ；来回迭代。这需要把数据多次扫过、内存/时间都大。

- **增量 EM (需要更多内存)**：

- 维护所有 s_i 与总和 μ 。
- 来一条数据：算新的 s_i^{new} ，用它替换旧的 s_i ($\mu \leftarrow \mu - s_i + s_i^{\text{new}}$)，据此立刻更新 $\theta(\mu)$ 。
- 优点：**单调收敛**；缺点：要保存**每个样本的** s_i （大数据时内存吃不消）。

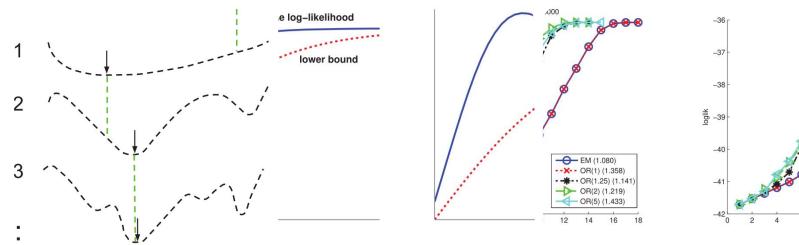
- **逐步 EM (常数内存)**：

- 不保存每个 s_i ，只保存 μ 。
- 来一条（或一个小批）数据：先算出 s_i ，再用**指数滑动平均**更新 $\mu \leftarrow (1 - \eta_k)\mu + \eta_k s_i$ （或小批的平均）。

- 步长 η_k 要符合 Robbins–Monro 条件 (大致: $\sum \eta_k = \infty$, $\sum \eta_k^2 < \infty$) , 常用 $\eta_k = (2 + k)^{-\kappa}$ 。
- 定好 μ 后再更新 $\theta(\mu)$ 。
- 优点: 内存 $O(1)$ 、适合流数据; 通过调 m, κ 能效率/稳定性兼顾。
- 经验结论 (按 Liang & Klein) :
 - 速度: 逐步 EM > 增量 EM >> 批量 EM。
 - 准确度: 逐步 EM \approx 甚至 > 批量 EM; 增量 EM 往往稍差。

要点小结

- 在线 EM 的两派:
 1. 增量 EM: 一次优化一个 q_i , 需要保存所有 s_i ; 单调收敛但内存大。
 2. 逐步 EM: 用随机逼近更新 μ : $\mu \leftarrow (1 - \eta_k)\mu + \eta_k \bar{s}$ (可用小批), 常数内存。
- 统一抽象:
 - $s_i = E[\phi(x_i, z) | x_i, \theta]$, $\mu = \sum_i s_i$, 参数更新写成 $\theta(\mu)$ 。
- 步长设置: $\eta_k = (2 + k)^{-\kappa}$, $0.5 < \kappa \leq 1$; 可用 minibatch 并在热身期并行试参。
- 实证: 逐步 EM 通常更快且精度不差, 非常适合大规模/流式学习场景。



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。

Figure 11.17 Illustration of deterministic annealing. Based on http://en.wikipedia.org/wiki/Graduated_optimization.

Figure 11.18 Illustration of possible behaviors of variational EM. (a) The lower bound increases at each iteration, and so does the likelihood. (b) The lower bound increases but the likelihood decreases. In this case, the algorithm is closing the gap between the approximate and true posterior. This can have a regularizing effect.

Based on Figure 6 of (Saul et al. 1996). Figure generated by varEMbound.

Figure 11.19 Illustration of adaptive over-relaxed EM applied to a mixture of 5 Gaussians in 15 dimensions. We show the algorithm applied to two different datasets, randomly sampled from a mixture of 10 Gaussians. We plot the convergence for different update rates η . Using $\eta = 1$ gives the same results as regular EM. The actual running time is printed in the legend. Figure generated by mixGaussOverRelaxedEmDemo.

11.4.9 Other EM variants

EM is one of the most widely used algorithms in statistics and machine learning. Not surprisingly, many variations have been proposed. We briefly mention a few below, some of which we will use in later chapters. See (McLachlan and Krishnan 1997) for more information.

- Annealed EM In general, EM will only converge to a local maximum. To increase the chance of finding the global maximum, we can use a variety of methods. One approach is to use a method known as deterministic annealing (Rose 1998). The basic idea is to “smooth” the posterior “landscape” by raising it to a temperature, and then gradually cooling it, all the while slowly tracking the global maximum. See Figure 11.17. for a sketch. (A stochastic version of this algorithm is described in Section 24.6.1.) An annealed version of EM is described in (Ueda and Nakano 1998).
- Variational EM In Section 11.4.7, we showed that the optimal thing to do in the E step is to make q_i be the exact posterior over the latent variables, $q_i(z_i) = p(z_i|x_i, \theta_t)$. In this case, the lower bound on the log likelihood will be tight, so the M step will “push up” on the log-likelihood itself. However, sometimes it is computationally intractable to perform exact inference in the E step, but we may be able to perform approximate inference. If we can ensure that the E step is performing inference based on a lower bound to the likelihood, then the M step can be seen as monotonically increasing this lower bound (see Figure 11.18). This is called variational EM (Neal and Hinton 1998). See Chapter 21 for some variational inference methods that can be used in the E step.
- Monte Carlo EM Another approach to handling an

intractable E step is to use a Monte Carlo approximation to the expected sufficient statistics. That is, we draw samples from the posterior, $z_{s i} \sim p(z_i | x_i, \theta_t)$, and then compute the sufficient statistics for each completed vector, $(x_i, z_{s i})$, and then average the results. This is called Monte Carlo EM or MCEM (Wei and Tanner 1990). (If we only draw a single sample, it is called stochastic EM (Celeux and Diebolt 1985).) One way to draw samples is to use MCMC (see Chapter 24). However, if we have to wait for MCMC to converge inside each E step, the method becomes very slow. An alternative is to use stochastic approximation, and only perform “brief” sampling in the E step, followed by a partial parameter update. This is called stochastic approximation EM (Delyon et al. 1999) and tends to work better than MCEM. Another alternative is to apply MCMC to infer the parameters as well as the latent variables (a fully Bayesian approach), thus eliminating the distinction between E and M steps. See Chapter 24 for details.

- Generalized EM Sometimes we can perform the E step exactly, but we cannot perform the M step exactly. However, we can still monotonically increase the log likelihood by performing a “partial” M step, in which we merely increase the expected complete data log likelihood, rather than maximizing it. For example, we might follow a few gradient steps. This is called the generalized EM or GEM algorithm. (This is an unfortunate term, since there are many ways to generalize EM....)
- ECM(E) algorithm The ECM algorithm stands for “expectation conditional maximization”, and refers to optimizing the parameters in the M step sequentially, if they turn out to be dependent. The ECME algorithm, which stands for “ECM either” (Liu and Rubin 1995), is a variant of ECM in which we maximize the expected complete data log likelihood (the Q function) as usual, or the observed data log likelihood, during one or more of the conditional maximization steps. The latter can be much faster, since it ignores the results of the E step, and directly optimizes the objective of interest. A standard example of this is when fitting the Student T distribution. For fixed v , we can update Σ as usual, but then to update v , we replace the standard update of the form $v_{t+1} = \arg \max_v Q((\mu_{t+1}, \Sigma_{t+1}, v), \theta_t)$ with $v_{t+1} = \arg \max_v \log p(D|$

$\mu_{t+1}, \Sigma_{t+1}, v$). See (McLachlan and Krishnan 1997) for more information.

- Over-relaxed EM Vanilla EM can be quite slow, especially if there is lots of missing data. The adaptive overrelaxed EM algorithm (Salakhutdinov and Roweis 2003) performs an update of the form $\theta_{t+1} = \theta_t + \eta(M(\theta_t) - \theta_t)$, where η is a step-size parameter, and $M(\theta_t)$ is the usual update computed during the M step. Obviously this reduces to standard EM if $\eta = 1$, but using larger values of η can result in faster convergence. See Figure 11.19 for an illustration. Unfortunately, using too large a value of η can cause the algorithm to fail to converge.

Finally, note that EM is in fact just a special case of a larger class of algorithms known as bound optimization or MM algorithms (MM stands for minorize-maximize). See (Hunter and Lange 2004) for further discussion.

下面把原文完整翻译；随后给出通俗解释与要点小结。

图注翻译

图 11.17 确定性退火 (deterministic annealing) 的示意图。改编自维基百科“Graduated optimization”。

图 11.18 变分 EM 可能出现的行为。(a) 每次迭代下界都上升，同时真实对数似然也上升。(b) 下界上升，但真实似然下降；此时算法在缩小“近似后验”和“真实后验”之间的差距，这会起到正则化作用。改编自 Saul 等 (1996) 图 6。

图 11.19 自适应“过松弛”EM (over-relaxed EM) 在 15 维、5 个高斯分量的混合上示意。对两个由 10 个高斯混合随机采样得到的数据集进行实验。横轴为迭代次数，纵轴为对数似然；对不同更新率 η 的收敛过程作图。取 $\eta = 1$ 与普通 EM 相同。图例中括号是实际运行时间。

正文翻译

11.4.9 其他 EM 变体

EM 是统计与机器学习中使用最广的算法之一。不出所料，人们提出了许多变体。下面简要提到几种（后续章节会用到其中一些）。更多信息见 McLachlan & Krishnan (1997)。

- **退火 EM (Annealed EM)**

一般来说，EM 只能收敛到**局部极大值**。为了增加找到全局极大的机会，可用多种方法。其中一种叫**确定性退火** (Rose, 1998)。基本思想是把后验的“地形”按一个“温度”进行**平滑**（抬幕或加温），然后**逐步降温**，同时缓慢跟踪全局极大点。见图 11.17。（其随机版本见 24.6.1 节。）Ueda & Nakano (1998) 给出了退火版 EM。

- **变分 EM (Variational EM)**

在 11.4.7 节我们说明过：E 步的最优选择是让 q_i 等于精确的后验 $q_i^t(z_i) = p(z_i | x_i, \theta_t)$ 。这样下界就**紧**，M 步会直接“顶起”真实对数似然。但有时**精确推断不可行**，可以做**近似推断**。若能保证 E 步使用的是**对似然的下界**（而非任意近似），则 M 步就能**单调提高这个下界**（见图 11.18），这称为**变分 EM** (Neal & Hinton, 1998)。E 步可用的变分推断方法见第 21 章。

- **蒙特卡洛 EM (Monte Carlo EM, MCEM)**

处理不可 tractable 的 E 步的另一法：用**蒙特卡洛**近似期望充分统计量。即从后验采样 $z_i^{(s)} \sim p(z_i | x_i, \theta_t)$ ，对每个“补全”的样本 $(x_i, z_i^{(s)})$ 计算充分统计量，再取平均 (Wei & Tanner, 1990)。若每次只采一个样本，称**随机 EM** (Celeux & Diebolt, 1985)。采样可用 MCMC (第 24 章)，但若每个 E 步都要等 MCMC 收敛，会很慢。替代方案是**随机逼近 EM** (Delyon 等, 1999)：在 E 步只做**短暂采样**，然后做**部分参数更新**；实践中通常比 MCEM 更好。另一条路是**全贝叶斯**：用 MCMC 同时推断参数与隐变量，从而不再区分 E/M 步（见第 24 章）。

- **广义 EM (Generalized EM, GEM)**

有时 E 步能精确做，但 M 步**无法精确最大化**。不过只要在 M 步**提高**（而非最大化）期望完备对数似然，也能**单调提升**目标；比如走几步梯度。这称为**广义 EM**。（名字有点“泛”，因为扩展 EM 的方式很多。）

- **ECM(E) 算法**

ECM = *Expectation–Conditional Maximization*：当参数之间有关联时，在 M 步**逐个**（或分块）条件最大化。**ECME** = *ECM Either* (Liu & Rubin, 1995)：在若干条件最大化步骤里，既可以最大化**Q 函数**（期望完备对数似然），也可以直接最大化**观测对数似然**。后者常更快，因为它忽略 E 步的中间量，直接对真正目标优化。一个标准示例是**拟合 Student-t**：固定 ν 时按常规更新 Σ ；更新 ν 时，不用 $\nu_{t+1} = \arg \max_\nu Q((\mu_{t+1}, \Sigma_{t+1}, \nu), \theta_t)$ ，而是 $\nu_{t+1} = \arg \max_\nu \log p(D | \mu_{t+1}, \Sigma_{t+1}, \nu)$ 。详见 McLachlan & Krishnan (1997)。

- **过松弛 EM (Over-relaxed EM)**

标准 EM (vanilla EM) 可能很慢，尤其数据缺失多时。**自适应过松弛 EM** (Salakhutdinov & Roweis, 2003) 采用

$$\theta_{t+1} = \theta_t + \eta(M(\theta_t) - \theta_t),$$

其中 $M(\theta_t)$ 是常规 M 步给出的更新， η 是步长； $\eta = 1$ 就是普通 EM，适当增大 η 可更快收敛（见图 11.19），但太大会不收敛。

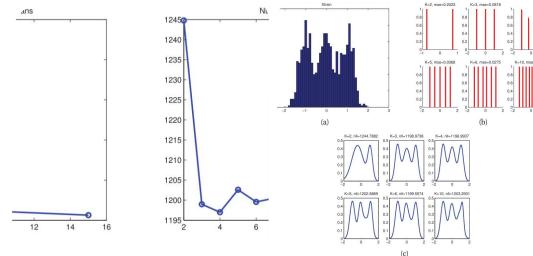
最后，注意 EM 实际上是更大一类算法——**下界优化/MM (minorize-maximize) **算法的特例。详见 Hunter & Lange (2004)。

通俗解释

- **退火 EM**：想象目标表面坑坑洼洼（很多局部最优）。把分布“加热”会把地形抹平，更容易往“全局高地”走；再慢慢降温、细化搜索。像爬山时先在薄雾中找大方向，再逐步清晰。
 - **变分 EM**：当 E 步的精确推断太贵时，用一个**近似的、但保证是下界**的后验来替代；于是每次都能保证**下界上升**。真实似然可能有时下降（图 11.18b），但“逼近更好”的效果常像**正则化**，防过拟合。
 - **MCEM / 随机逼近 EM**：用采样来估期望。MCEM 要把采样做“准到位”，会慢；随机逼近 EM 每步只“采一点、更新一点”，往往更高效、也更稳。
 - **GEM**：M 步做不到最优没关系——**只要比上一步好**就行；比如走几步梯度、坐标上升等。
 - **ECM/ECME**：M 步拆成**多块**依次优化；有时直接对**真实对数似然**优化一部分比对 Q 函数更快 (t 分布的 v 就是例子)。
 - **过松弛 EM**：把“EM 的步子”放大一点 ($\eta > 1$) 能更快，但太大胆会摔跤（发散）；通常需要**自适应**地调 η 。
 - **大图景**：EM 属于“**下界优化 (MM)**”——先找个**容易最大化的**下界贴在原目标上，再在下界上做改进。
-

要点小结

- EM 的多种加速/稳健化方案：**退火、变分、蒙特卡洛/随机逼近、GEM、ECM/ECME、过松弛**。
- 变分 EM：保证**下界单调上升**；真实似然不一定每步上升，但常带来**正则化**。
- MCEM 慢；**随机逼近 EM**更实用。
- **过松弛**可加速但有发散风险；需调步长 η 。
- EM 是 MM/**下界优化**的一个特例——理解这一点有助于设计新的变体与收敛性分析。



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。

Figure 11.20 Test set performance vs K for data generated from a mixture of 3 Gaussians in 1d (data is shown in Figure 11.21(a)). (a) MSE on test set for K-means. (b) Negative log likelihood on test set for GMM. Figure generated by kmeansModelSel1d.

Figure 11.21 Synthetic data generated from a mixture of 3 Gaussians in 1d. (a) Histogram of training data. (Test data looks essentially the same.) (b) Centroids estimated by K-means for $K \in \{2, 3, 4, 5, 6, 10\}$. (c) GMM density model estimated by EM for for the same values of K. Figure generated by kmeansModelSel1d.

11.5 Model selection for latent variable models

When using LVMs, we must specify the number of latent variables, which controls the model complexity. In particular, in the case of mixture models, we must specify K , the number of clusters. Choosing these parameters is an example of model selection. We discuss some approaches below.

11.5.1 Model selection for probabilistic models

The optimal Bayesian approach, discussed in Section 5.3, is to pick the model with the largest marginal likelihood, $K^* = \text{argmax}_K p(D|K)$. There are two problems with this.

First, evaluating the marginal likelihood for LVMs is quite difficult. In practice, simple approximations, such as BIC, can be used (see e.g., (Fraley and Raftery 2002)).

Alternatively, we can use the cross-validated likelihood as a performance measure, although this can be slow, since it requires fitting each model F times, where F is the number of CV folds.

The second issue is the need to search over a potentially large number of models. The usual approach is to perform

exhaustive search over all candidate values of K. However, sometimes we can set the model to its maximal size, and then rely on the power of the Bayesian Occam's razor to "kill off" unwanted components. An example of this will be shown in Section 21.6.1.6, when we discuss variational Bayes. An alternative approach is to perform stochastic sampling in the space of models. Traditional approaches, such as (Green 1998, 2003; Lunn et al. 2009), are based on reversible jump MCMC, and use birth moves to propose new centers, and death moves to kill off old centers.

However, this can be slow and difficult to implement. A simpler approach is to use a Dirichlet process mixture model, which can be fit using Gibbs sampling, but still allows for an unbounded number of mixture components; see Section 25.2 for details. Perhaps surprisingly, these sampling-based methods can be faster than the simple approach of evaluating the quality of each K separately. The reason is that fitting the model for each K is often slow. By contrast, the sampling methods can often quickly determine that a certain value of K is poor, and thus they need not waste time in that part of the posterior.

11.5.2 Model selection for non-probabilistic methods

What if we are not using a probabilistic model? For example, how do we choose K for the Kmeans algorithm? Since this does not correspond to a probability model, there is no likelihood, so none of the methods described above can be used. An obvious proxy for the likelihood is the reconstruction error. Define the squared reconstruction error of a data set D, using model complexity K, as follows:

$$E(D,K) = \frac{1}{|D|} \sum_{i \in D} \|x_i - \hat{x}_i\|^2 \quad (11.95)$$

In the case of K-means, the reconstruction is given by $\hat{x}_i = \mu_{z_i}$, where $z_i = \arg \min_k \|x_i - \mu_k\|^2$, as explained in Section 11.4.2.6. Figure 11.20(a) plots the reconstruction error on the test set for K-means. We notice that the error decreases with increasing model complexity! The reason for this behavior is as follows:

when we add more and more centroids to K-means, we can "tile" the space more densely, as shown in Figure 11.21(b). Hence any given test vector is more likely to find a close prototype to accurately represent it as K increases, thus decreasing reconstruction error. However, if we use a probabilistic model, such as the GMM, and plot the

negative log-likelihood, we get the usual U-shaped curve on the test set, as shown in Figure 11.20(b). In supervised learning, we can always use cross validation to select between non-probabilistic models of different complexity, but this is not the case with unsupervised learning. Although this is not a novel observation (e.g., it is mentioned in passing in (Hastie et al. 2009, p519), one of the standard references in this field), it is perhaps not as widely appreciated as it should be. In fact, it is one of the more compelling arguments in favor of probabilistic models. Given that cross validation doesn't work, and supposing one is unwilling to use probabilistic models (for some bizarre reason...), how can one choose K? The most common approach is to plot the reconstruction error on the training set versus K, and to try to identify a knee or kink in the curve. The idea is that for $K < K_*$, where K_* is the "true" number of clusters, the rate of decrease in the error function will be high, since we are splitting apart things that should not be grouped together. However, for $K > K_*$, we are splitting apart "natural" clusters, which does not reduce the error by as much. This kink-finding process can be automated by use of the gap statistic (Tibshirani et al. 2001). Nevertheless, identifying such kinks can be hard, as shown in Figure 11.20(a), since the loss function usually drops off gradually. A different approach to "kink finding" is described in Section 12.3.2.1.

下面把原文完整翻译成中文；随后用通俗方式解释，并给出要点小结。

图注翻译

图 11.20 一维三高斯混合生成的数据（训练/测试数据如图 11.21(a)）上，测试集性能随 K 的变化。 (a) K-means 的测试集 MSE。 (b) GMM 的测试集负对数似然 (NLL)。图由 `kmeansModelSel1d` 生成。

图 11.21 一维三高斯混合的合成数据。 (a) 训练数据的直方图。（测试集外观几乎一样。）(b) 当 $K \in \{2, 3, 4, 5, 6, 10\}$ 时，K-means 估计得到的质心（纵轴只是显示用的 1）。(c) 对同样的 K ，用 EM 估计得到的 GMM 密度模型。图由 `kmeansModelSel1d` 生成。

正文翻译

11.5 潜变量模型的模型选择

使用潜变量模型 (LVM) 时，必须指定**潜变量的个数**，这会控制模型复杂度。特别地，对混合模型，需要指定**簇的个数** K 。选择这些参数就是**模型选择**的一个例子。下面讨论几种方法。

11.5.1 概率模型的模型选择

最优的贝叶斯做法（见 5.3 节）是选择**边际似然**最大的模型：

$$K^* = \arg \max_K p(D | K).$$

这有两个问题。第一，对 LVM 计算边际似然很困难。实际中可用简单近似，如 BIC（例如 Fraley & Raftery, 2002）。另一种是把**交叉验证的似然**作为性能指标，但这会很慢，因为需要对每个模型做 F 次拟合 (F 是折数)。

第二个问题是要在可能很大的模型集合里搜索。通常的做法是对所有候选的 K 穷举搜索。但有时我们可以把模型设到一个**最大规模**，然后依赖贝叶斯的**奥卡姆剃刀**自动“砍掉”不需要的成分。第 21.6.1.6 节在讲变分贝叶斯时会看到一个例子。另一个办法是在**模型空间**里做随机采样。传统方法如可逆跳跃 MCMC (Green 1998, 2003; Lunn 等 2009)：通过“出生”提议新中心，通过“死亡”去掉旧中心。但这既慢又难实现。更简单的是用**狄利克雷过程混合 (DPMM)**，它可以用 Gibbs 采样拟合，同时允许**无上界**的分量数；见第 25.2 节。可能令人惊讶的是，这些基于采样的方法有时比“逐个 K 评估”的朴素方法更快。原因是：对每个 K 完整拟合往往很慢，而采样方法能很快判断某些 K 很差，从而不在该部分后验上浪费时间。

11.5.2 非概率方法的模型选择

如果我们**不用概率模型**怎么办？例如，K-means 中如何选 K ？由于它不是概率模型，没有似然，前面的办法都用不上。一个显而易见的替代指标是**重建误差**。定义使用复杂度为 K 的模型对数据集 D 的平方重建误差：

$$E(D, K) = \frac{1}{|D|} \sum_{i \in D} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 \quad (11.95)$$

对于 K-means，重建 $\hat{\mathbf{x}}_i = \mu_{z_i}$ ，其中

$$z_i = \arg \min_k \|\mathbf{x}_i - \mu_k\|_2^2 \quad (\text{见 } 11.4.2.6 \text{ 节}) .$$

图 11.20(a) 画出了 K-means 在测试集上的重建误差。可以看到**误差随 K 增大而下降！**原因是：当我们不断增加质心数，便可以如图 11.21(b) 那样把空间**更密集地铺瓷砖**，因此任何给定的测试向量都更容易找到一个很近的原型来表示，误差自然降低。

然而，如果使用**概率模型**（如 GMM），并画出测试集的**负对数似然**，就会得到熟悉的 U 形曲线（图 11.20(b)）： K 太小欠拟合，似然低； K 过大过拟合，测试似然又下降。

在**监督学习**里，我们总能用交叉验证来在不同复杂度的**非概率模型**之间做选择，但在**无监督学习**里并不行。虽然这并非新观点（例如 Hastie 等 2009 第 519 页提到过），但它并没有被足够重视——这也是支持**概率模型**的一个有力论据。

既然交叉验证不好用，而你又（不知为何）不想用概率模型，那该怎么选 K ？最常见做法是：画出**训练集**上的重建误差随 K 的曲线，试图找到一个“**拐点/折点**”（elbow）。直觉是：当 $K < K^{*}$ (K^{*} 为“真正的”簇数)，误差会快速下降，因为我们在把不该合在一起的东西拆开；而当 $K > K^{*}$ 时，我们是在把“自然的簇”继续细分，这不会显著降低误差。这个“找拐点”的过程可以用 **gap 统计量** (Tibshirani 等 2001) 自动化。不过，如图 11.20(a) 所示，损失函数通常是**平滑逐渐下降**，拐点并不明显，因此自动化也不容易。另一种“找拐点”的思路见 12.3.2.1 节。

通俗解释

- **为什么 GMM 的测试 NLL 是 U 形？**

K 小：模型太粗，拟合不了数据结构 → 欠拟合，测试 NLL 大。

K 中等：刚好合适 → 测试 NLL 最小。

K 大：参数太多，把训练噪声也“记住了” → 过拟合，测试 NLL 反而变差。

- **为什么 K-means 的测试误差一直降？**

K-means 只是“用最近的原型来量化数据”的**压缩器**。多放几个原型，总能离任何点更近一些，所以**重建误差必然下降**。但这并不代表泛化更好——它只是更会“记住形状”，没有“惩罚复杂度”的机制。

- **如何为 LVM 选 K ？**

- 概率方法：边际似然（或 BIC/变分下界/交叉验证似然）。

- 全贝叶斯：RJMCMC、DPMM 等在模型空间里采样，能自动决定成分数。

- 非概率方法（如 K-means）：用训练重建误差画“**肘部法则**”，或用 **gap 统计量** 辅助，但拐点并不总清晰。

- **为什么有时用采样比“逐个 K 训练”更快？**

逐个 K 都要把模型训练到收敛；而采样法能在游走过程中迅速发现“不合适的 K ”，从而少花冤枉功夫。

要点小结

- LVM 需要选择复杂度（如混合模型的 K ）。

- **概率模型可用**: 边际似然/BIC、交叉验证似然、变分下界; 也可用采样 (RJMCMC, DPMM) 在模型空间探索。
- **非概率模型** (如 K-means) 没有似然, 常用**重建误差 + 肘部法/Gap 统计量**; 但误差通常随 K 单调下降, 拐点不总明显。
- GMM 在测试集上的 NLL 通常呈 U 形; K-means 的重建误差通常单调下降。
- 这说明: 在无监督学习里, 若要稳健地做模型选择, **概率建模更有优势**。

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。

11.6 Fitting models with missing data

Suppose we want to fit a joint density model by maximum likelihood, but we have “holes” in our data matrix, due to missing data (usually represented by NaNs). More formally, let $O_{ij} = 1$ if component j of data case i is observed, and let $O_{ij} = 0$ otherwise. Let $X_v = \{x_{ij} : O_{ij} = 1\}$ be the visible data, and $X_h = \{x_{ij} : O_{ij} = 0\}$ be the missing or hidden data. Our goal is to compute

$$\hat{\theta} = \operatorname{argmax}_{\theta} p(X_v | \theta, O) \quad (11.96)$$

Under the missing at random assumption (see Section 8.6.2), we have

$$p(X_v | \theta, O) = \prod_{i=1}^N p(x_{iv} | \theta) \quad (11.97)$$

where x_{iv} is a vector created from row i and the columns indexed by the set $\{j : O_{ij} = 1\}$. Hence the log-likelihood has the form

$$\log p(X_v | \theta) = \sum_{i=1}^N \log p(x_{iv} | \theta) \quad (11.98)$$

where

$$p(x_{iv} | \theta) = \prod_{h \in H_i} p(x_{ih} | \theta) \quad (11.99)$$

and x_{ih} is the vector of hidden variables for case i (assumed discrete for notational simplicity). Substituting in, we get

$$\log p(X_v | \theta) = \sum_{i=1}^N \log \left(\prod_{h \in H_i} p(x_{ih} | \theta) \right) \quad (11.100)$$

Unfortunately, this objective is hard to maximize. since we cannot push the log inside the sum. However, we can use the EM algorithm to compute a local optimum. We give an example of this below.

11.6.1 EM for the MLE of an MVN with missing data

Suppose we want to fit an MVN by maximum likelihood,

but we have missing data. We can use EM to find a local maximum of the objective, as we explain below.

11.6.1.1 Getting started

To get the algorithm started, we can compute the MLE based on those rows of the data matrix that are fully observed. If there are no such rows, we can use some ad-hoc imputation procedures, and then compute an initial MLE.

11.6.1.2 E step Once we have θ_{t-1} , we can compute the expected complete data log likelihood at iteration t as follows:

$$Q(\theta, \theta_{t-1}) = E \sum_i \log N(x_i | \mu, \Sigma) | D, \theta_{t-1} \quad (11.101)$$

$$= -N/2 \log |2\pi\Sigma| - 1/2 \sum_i (x_i - \mu)^T \Sigma^{-1} (x_i - \mu) \quad (11.102)$$

$$= -N/2 \log |2\pi\Sigma| - 1/2 \text{tr}(\Sigma^{-1} \sum_i (x_i - \mu)(x_i - \mu)^T) \quad (11.103)$$

$$= -N/2 \log |\Sigma| - ND/2 \log(2\pi) - 1/2 \text{tr}(\Sigma^{-1} E[S(\mu)]) \quad (11.104)$$

where

$$E[S(\mu)] = \sum_i E[x_i x_i^T] + \mu \mu^T - 2\mu E[x_i] \quad (11.105)$$

(We drop the conditioning of the expectation on D and θ_{t-1} for brevity.) We see that we need to compute $\sum_i E[x_i]$ and $\sum_i E[x_i x_i^T]$; these are the expected sufficient statistics.

To compute these quantities, we use the results from Section 4.3.1. Specifically, consider case i, where components v are observed and components h are unobserved. We have

$$x_{ih}|x_{iv}, \theta \sim N(m_i, V_i) \quad (11.106)$$

$$m_i = \mu_h + \Sigma_{hv} \Sigma^{-1} v v^T (x_{iv} - \mu_v) \quad (11.107)$$

$$V_i = \Sigma_{hh} - \Sigma_{hv} \Sigma^{-1} v v^T \Sigma_{vh} \quad (11.108)$$

Hence the expected sufficient statistics are

$$E[x_i] = (E[x_{ih}]; x_{iv}) = (m_i; x_{iv}) \quad (11.109)$$

where we have assumed (without loss of generality) that the unobserved variables come before the observed variables in the node ordering. To compute $E[x_i x_i^T]$, we use the result that $\text{cov}[x] = E[xx^T] - E[x]E[x^T]$. Hence

$$E[x_i x_i^T] = E[x_{ih} x_{iv} x_{iv}^T x_{ih}^T] = E[x_{ih} x_{ih}^T] + E[x_{ih} x_{ih}^T x_{iv} x_{iv}^T] \quad (11.110)$$

$$E[x_{ih} x_{ih}^T] = E[x_{ih}] E[x_{ih}]^T + V_i \quad (11.111)$$

11.6.1.3 M step

By solving $\nabla Q(\theta, \theta_{t-1}) = 0$, we can show that the M step is equivalent to plugging these ESS into the usual MLE equations to get

$$\mu_t = 1/N \sum_i E[x_i] \quad (11.112)$$

$$\Sigma_t = 1/N \sum_i E[x_i x_i^T] - \mu_t (\mu_t)^T \quad (11.113)$$

Thus we see that EM is not equivalent to simply replacing variables by their expectations and applying the standard MLE formula; that would ignore the posterior variance and would result in an incorrect estimate. Instead we must compute the expectation of the sufficient statistics, and plug that into the usual equation for the MLE. We can easily modify the algorithm to perform MAP estimation, by plugging in the ESS into the equation for the MAP estimate. For an implementation, see gaussMissingFitEm.

11.6.1.4 Example

As an example of this procedure in action, let us reconsider the imputation problem from Section 4.3.2.3, which had $N = 100$ 10-dimensional data cases, with 50% missing data. Let us fit the parameters using EM. Call the resulting parameters $\hat{\theta}$. We can use our model for predictions by computing $E[x_{ih}|x_{iv}, \hat{\theta}]$. Figure 11.22(a-b) indicates that the results obtained using the learned parameters are almost as good as with the true parameters. Not surprisingly, performance improves with more data, or as the fraction of missing data is reduced.

11.6.1.5 Extension to the GMM case

It is straightforward to fit a mixture of Gaussians in the presence of partially observed data vectors x_i . We leave the details as an exercise.

已思考若干秒

下面先把原文逐句不漏翻译成中文；随后给出通俗解释和要点总结，尽量不用太多高深数学。

翻译

11.6 含缺失数据时的模型拟合

假设我们想用极大似然来拟合一个联合密度模型，但数据矩阵里有“洞”（缺失值，通常用 NaN 表示）。更形式化地，若第 i 条样本的第 j 个分量被观测到，则令 $O_{ij} = 1$ ；否则 $O_{ij} = 0$ 。记可见数据为 $X_v = \{x_{ij} : O_{ij} = 1\}$ ，缺失/隐藏数据为 $X_h = \{x_{ij} : O_{ij} = 0\}$ 。我们的目标是计算

$$\hat{\theta} = \arg \max_{\theta} p(X_v | \theta, O) \quad (11.96)$$

在“随机缺失 (MAR)”假设下 (见 8.6.2 节)，有

$$p(X_v \mid \theta, O) = \prod_{i=1}^N p(x_i^v \mid \theta) \quad (11.97)$$

其中 x_i^v 是由第 i 行中被观测到的那些列所组成的向量（这些列由集合 $\{j : O_{ij} = 1\}$ 指定）。因此对数似然为

$$\log p(X_v \mid \theta) = \sum_i \log p(x_i^v \mid \theta) \quad (11.98)$$

并且

$$p(x_i^v \mid \theta) = \sum_{x_i^h} p(x_i^v, x_i^h \mid \theta) \quad (11.99)$$

这里 x_i^h 是第 i 条样本中缺失变量所成的向量（为简化记号，假设它是离散的）。代入得

$$\log p(X_v \mid \theta) = \sum_i \log \sum_{x_i^h} p(x_i^v, x_i^h \mid \theta) \quad (11.100)$$

不幸的是，这个目标函数很难最大化，因为我们不能把 \log 推进求和号。不过，我们可以用 EM 算法来求一个局部最优。下面给出一个例子。

11.6.1 用 EM 求带缺失数据的多元高斯 (MVN) 的 MLE

假设我们要对一个 MVN 做极大似然拟合，但存在缺失数据。可以用 EM 来寻找该目标的局部极大值，方法如下。

11.6.1.1 初始化

为了启动算法，我们可以先用数据矩阵中完全观测的那些行来计算 MLE。如果一个完全观测的行都没有，可以先做一些临时的插补，然后算一个初始的 MLE。

11.6.1.2 E 步

一旦有了 θ_{t-1} ，在第 t 次迭代，我们计算期望的完备数据对数似然：

$$Q(\theta, \theta^{t-1}) = E \left[\sum_{i=1}^N \log N(x_i \mid \mu, \Sigma) \mid D, \theta^{t-1} \right] \quad (11.101)$$

$$= -\frac{N}{2} \log |2\pi\Sigma| - \frac{1}{2} \sum_i E[(x_i - \mu)^\top \Sigma^{-1} (x_i - \mu)] \quad (11.102)$$

$$= -\frac{N}{2} \log |2\pi\Sigma| - \frac{1}{2} \text{tr}(\Sigma^{-1} \sum_i E(x_i - \mu)(x_i - \mu)^\top) \quad (11.103)$$

$$= -\frac{N}{2} \log |\Sigma| - \frac{ND}{2} \log(2\pi) - \frac{1}{2} \text{tr}(\Sigma^{-1} E[S(\mu)]) \quad (11.104)$$

其中

$$E[S(\mu)] = \sum_i (E[x_i x_i^\top] + \mu \mu^\top - 2\mu E[x_i]^\top) \quad (11.105)$$

(为简洁起见, 在期望符号上省略了对 D, θ^{t-1} 的条件。)

可见我们需要计算 $\sum_i E[x_i]$ 和 $\sum_i E[x_i x_i^\top]$, 这就是**期望充分统计量** (ESS)。计算它们时, 用 4.3.1 节的结果: 对第 i 个样本, 设可见分量为 v , 缺失分量为 h , 则

$$x_i^h | x_i^v, \theta \sim N(m_i, V_i) \quad (11.106)$$

$$m_i = \mu_h + \Sigma_{hv} \Sigma_{vv}^{-1} (x_i^v - \mu_v) \quad (11.107)$$

$$V_i = \Sigma_{hh} - \Sigma_{hv} \Sigma_{vv}^{-1} \Sigma_{vh} \quad (11.108)$$

于是

$$E[x_i] = (E[x_i^h]; x_i^v) = (m_i; x_i^v) \quad (11.109)$$

(不失一般性地, 假定未观测变量在节点排序中位于已观测变量之前。)

再利用 $\text{cov}[x] = E[xx^\top] - E[x]E[x]^\top$, 得

$$E[x_i x_i^\top] = \begin{bmatrix} E[x_i^h x_i^{h\top}] & E[x_i^h] x_i^{v\top} \\ x_i^v E[x_i^h]^\top & x_i^v x_i^{v\top} \end{bmatrix} \quad (11.110)$$

$$E[x_i^h x_i^{h\top}] = E[x_i^h] E[x_i^h]^\top + V_i \quad (11.111)$$

11.6.1.3 M 步

解 $\nabla Q(\theta, \theta^{t-1}) = 0$ 可得, M 步等价于把这些 ESS 带入常规 MLE 公式:

$$\mu^t = \frac{1}{N} \sum_i E[x_i] \quad (11.112)$$

$$\Sigma^t = \frac{1}{N} \sum_i E[x_i x_i^\top] - \mu^t (\mu^t)^\top \quad (11.113)$$

所以, EM 不是简单地“把缺失项用其期望替换后, 再套用普通 MLE”, 那样会忽略后验方差, 得到错误估计; 必须计算**充分统计量的期望**, 再代入标准的 MLE 公式。若要做 MAP, 只需把 ESS 代入 MAP 公式即可。实现可见 `gaussMissingFitEm`。

11.6.1.4 例子

作为该过程的一个实例，回到 4.3.2.3 节的插补问题： $N = 100$ 、10 维数据、50% 缺失。用 EM 拟合参数，得到 $\hat{\theta}$ 。预测时计算 $E[x_i^h | x_i^v, \hat{\theta}]$ 。图 11.22(a-b) 显示：用学得的参数得到的结果几乎和用真实参数一样好。显然，数据越多或缺失比例越小，表现越好。

11.6.1.5 推广到 GMM

当样本向量 x_i 只有部分分量被观测到时，拟合高斯混合也很直接；细节留作练习。

通俗解释（少数学）

- **问题：**数据表里有很多空值。我们想拟合一个（比如高斯）联合分布，但似然里要对“缺失项”求和（或积分），而 \log 不能进求和号 → 直接最大化很难。
 - **EM 的主意：**
 - **E 步：**在当前参数下，给每条样本的“缺失部分”算出**条件分布**（对高斯就是条件高斯），取得**条件均值** m_i 和**条件协方差** V_i 。据此累加两样“账本”： $\sum E[x_i]$ 与 $\sum E[x_i x_i^\top]$ 。注意：不仅要记均值，也要把**方差** V_i 算进去。
 - **M 步：**把这两本“期望账”当作完整数据的充分统计量，直接用**普通 MLE 公式**更新 μ 与 Σ 。
 - 反复迭代直到收敛。
 - **为什么不能只用均值替换：**如果你把缺失项直接填成 $E[x_i^h]$ 再算协方差，会把缺失的不确定性当作 0，从而**低估协方差**。EM 通过把 V_i 加进 $E[x_i x_i^\top]$ 里，正确地“记账”了这份不确定性。
 - **初始化：**先用**完整行**求个初值；没有完整行就先随便插补一下（均值/回归等），再启动 EM。
 - **GMM 的扩展：**在上面基础上再加“簇责任度” r_{ik} ：E 步同时算条件高斯 + 责任度，M 步对每个簇用加权的 ESS 更新即可。
-

实操小抄 (MVN + 缺失)

1. **初始化** μ, Σ ：用完整行；或粗略插补后求一次。
2. **E 步**（对每条样本 i ）：
 - 划分观测 v 和缺失 h 。
 - 计算

$$m_i = \mu_h + \Sigma_{hv} \Sigma_{vv}^{-1} (x_i^v - \mu_v), \quad V_i = \Sigma_{hh} - \Sigma_{hv} \Sigma_{vv}^{-1} \Sigma_{vh}.$$

- 累加: $E[x_i] = [m_i; x_i^v]$,
 $E[x_i x_i^\top] = \begin{bmatrix} m_i m_i^\top + V_i & m_i x_i^{v\top} \\ x_i^v m_i^\top & x_i^v x_i^{v\top} \end{bmatrix}$.

3. M 步:

$$\mu \leftarrow \frac{1}{N} \sum_i E[x_i], \quad \Sigma \leftarrow \frac{1}{N} \sum_i E[x_i x_i^\top] - \mu \mu^\top.$$

4. 收敛判定: 对数似然或参数变化小于阈值即停; 做 MAP 时在 M 步里加先验 (如 NIW)。
-

要点小结

- 缺失值下的目标是观测数据对数似然; log 外面有求和, 直接最大化难 → 用 EM。
- E 步要计算条件均值和条件协方差, 形成期望充分统计量 (不仅是均值, 还包括二阶矩)。
- M 步把 ESS 代回标准 MLE (或 MAP) 公式更新参数。
- EM ≠ “简单均值填充后做 MLE”, 因为那会忽视后验方差。
- 初始化可用完整行或粗插补; 数据越多、缺失越少, 效果越好。
- 方法可直接推广到 GMM (再加簇责任度)。

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 28s

下面把整章要点打包成三部分给你: ①知识点总结 ②子章节关联 ③本章总结。内容只保留“干货+用途”, 便于复习与落地。

① 知识点总结 (分点 + 关键词 + 用途)

- 潜变量模型 LVM (11.1)
 - 关键词: hidden/latent、bottleneck、压缩表示、依赖结构、DGM。
 - 要点: 用隐藏变量解释可见变量的相关性; 参数更少、可做无监督表示学习; 常见结构: 多对多/一对多/多对一/一对一。

- 用途：降维 (PCA/FA/ICA) 、主题/成分建模、生成式表示。
- **混合模型 Mixture models (11.2)**
 - 关键词：离散潜变量 $z \sim \text{Cat}(\pi)$ 、凸组合、base densities。
 - 要点：总体密度 $p(x) = \sum_k \pi_k p_k(x)$ 。
 - 用途：密度估计、异常检测、生成式分类器 $p(x|y)$ 、聚类（软/硬）。
- **高斯混合 GMM (11.2.1)**
 - 关键词： $N(\mu_k, \Sigma_k)$ 、可逼近任意连续密度。
 - 用途：连续数据密度/聚类、图像/语音建模。
- **伯努利/多项式混合 (11.2.2)**
 - 关键词：Prod. Bernoulli/Multinoulli、像素/词袋独立。
 - 要点：虽然分量内独立，但**混合后能产生相关性**。
 - 用途：二值图像/文本的聚类与建模。
- **混合的专家 MoE (11.2.4)**
 - 关键词：gating $p(z|x) = \text{softmax}(V^\top x)$ 、专家 $p(y|x, z = k)$ 、条件混合。
 - 要点：分区分片的回归/分类；可层次化 (HME)；亦可用 NN (Mixture Density Network)。
 - 用途：分段函数拟合、复杂决策面、**逆问题**（多模态输出时用“众数”而非“均值”）。
- **参数估计难点 (11.3)**
 - 关键词：不可识别 (label switching)、多峰、非凸 (log-sum-exp)、局部最优。
 - 用途：指导初始化/多次重启/选 MAP 而非纯 ML。
- **EM 算法 (11.4 核心)**
 - 关键词：E 步 (算期望充分统计量 ESS)、M 步 (最大化 Q)、单调不降、MAP=加上 $\log p(\theta)$ 。
 - 用途：带潜变量或缺失数据的 ML/MAP 估计的“万能工作马”。
- **EM—GMM 实例 (11.4.2)**
 - 关键词：责任度 r_{ik} 、更新 π, μ, Σ 。
 - 变体：K-means=球协方差+等权+硬分配 (hard EM)；VQ=压缩视角；初始化（随机、k-means++、增量分裂）；
 - 稳定性：ML 的“方差塌缩”奇异点，MAP 用 Dirichlet+NIW 先验可避免。
- **EM—MoE (11.4.3)**

- 关键词：E：软责任；M：加权最小二乘求 w_k, σ_k^2 ，对 gating 做软标签逻辑回归。
- 用途：高效拟合 MoE/HME。
- EM—带隐藏变量的 DGM (11.4.4)
 - 关键词：家族边缘、期望计数 N_{tck} 、表格 CPD。
 - 用途：一般图模型的参数学习；MAP=加伪计数。
- EM—Student t (11.4.5)
 - 关键词：高斯尺度混合、辅助 $z \sim \text{Gamma}$ 、鲁棒、 ν 一维优化 (GEM/ECME)。
 - 用途：抗离群点的密度/分类；“破产数据”示例显著优于高斯。
- EM—Probit 回归 (11.4.6)
 - 关键词：潜变量 z 截断高斯；E: $E[z|y, x]$ ；M: 岭回归解。
 - 用途：二分类的另一条路；虽慢于拟牛顿，但可推广到更复杂模型。
- EM 的理论 (11.4.7)
 - 关键词：下界/ELBO、Jensen、不等式、E 步选 $q = p(z|x, \theta)$ 使下界“贴住”原函数→单调提升。
 - 用途：调试（若似然下降=有 bug）、变分/MC/在线 EM 的基石。
- 在线 EM (11.4.8)
 - 关键词：增量 EM（存每例 ESS，单调）、逐步/随机逼近 EM（只存全局、步长 η_k 、可小批）。
 - 用途：大数据/流式学习；实践上 stepwise 更快。
- 其他 EM 变体 (11.4.9)
 - 关键词：退火 EM、变分 EM（近似 E 步提升下界）、MCEM/SAEM、GEM、ECM/ECME、过松弛 EM（加步长 $\eta > 1$ ）。
 - 用途：跳出局部最优、应对难优化 M 步、加速收敛。
- 模型选择 (11.5)
 - 概率法：边际似然/BIC/交叉验证、RJMCMC、DPMM（不用预设 K ）。
 - 非概率 (K-means)：用重建误差，通常随 K 单调下降；“肘部/Gap 统计量”找拐点。
 - 要点：无监督里，概率模型更利于模型选择（测试 NLL 呈 U 形）。
- 缺失数据的拟合 (11.6)
 - 关键词：MAR；MVN 条件分布 $x_h|x_v$ 的均值/协方差；
 - E: 算 $E[x], E[xx^\top]$ (含后验方差 V_i)；M: 常规 MLE 公式。

- 用途：带 NaN 的高斯/高斯混合学习与插补；切记不是“均值填充”。

② 子章节关联（依赖 / 对比 / 递进）

• 依赖

- 11.1 (LVM 概念) \Rightarrow 11.2 (混合模型是最简单的 LVM)。
- 11.3 (估计难点：多峰/非凸) \Rightarrow 11.4 (提出 EM 作为主力解法)。
- 11.4.7 (EM 单调理论) \Rightarrow 支撑 11.4.8 (在线) 与 11.4.9 (变体) 的正确性。
- 11.6 (缺失数据) 依赖 11.4 的 E/M 思想与 4.3 的条件高斯公式。

• 对比

- K-means vs GMM：硬 vs 软、非概率 vs 概率、重建误差单调降 vs 测试 NLL U 形。
- ML vs MAP：ML 有奇异/过拟合风险；MAP 用先验稳健，数值更可靠。
- 均值预测 vs 众数预测（逆问题）：均值差、众数合理（多峰输出）。

• 递进

- GMM \rightarrow MoE/HME (权重从常数到输入依赖) \rightarrow Mixture Density Network (NN 版)。
- 批量 EM \rightarrow 增量/随机 EM (规模化) \rightarrow 退火/过松弛/变分/MC (更强鲁棒/更快)。

(可选 Mermaid 关系图)

```
graph TD; A11_1[LVM 概念] --> B11_2[混合模型]; B11_2 --> C11_3[参数估计难点: 不可识别/非凸]; C11_3 --> D11_4[EM 算法]; D11_4 --> D2[EM 实例: GMM/MoE/DGM/Student/Probit]; D11_4 --> D3[EM 理论: 下界与单调]; D3 --> D4[在线EM/变体]; D11_4 --> E11_6[缺失数据的EM]; D11_4 --> F11_5[模型选择(选K)]
```

③ 本章总结（一屏速记）

- 主线：**本章围绕“**潜变量 + 非凸似然**”展开，提出 EM 作为通用求解框架：
E 步用当前参数对隐藏量（或缺失值/辅助变量）**求条件期望 (ESS)**；M 步在“**填充后的充分统计量**”上**做封闭式更新或部分优化**。EM 在理论上是**下界最大化**，可保证目标（或下界）**单调不降**。
- 模型家族：**从**混合模型** (GMM/伯努利) 到**混合的专家、带隐藏结点的 DGM**、**Student t/Probit** 等，都可落地到 EM/GEM/ECME 形式；MoE 解决输入条件下的**多模态映射**；Student t 对离群点更鲁棒。

- **实践要点：**

1. **初始化关键：**K-means++、多重重启、增量分裂。
 2. **用 MAP 避免 GMM 的“方差塌缩”，数值更稳。**
 3. **选择 K：**概率模型用 BIC/交叉验证/贝叶斯非参（DPMM）；非概率模型只能“肘部/Gap”。
 4. **大规模：**用增量/随机（stepwise）EM、小批次与步长策略。
 5. **调试：**似然不升就有 bug；必要时用退火、过松弛、变分/MCEM 加速或逃逸局部最优。
 6. **缺失值：**别直接均值填充——要把**均值+方差**都计入 ESS 再更新。
- **一句话：**EM 是潜变量世界的瑞士军刀——把“看不见的”先用概率“补”出来，再在补好的充分统计量上做稳定更新；配合合适的先验、初始化与模型选择，就能在密度估计、聚类、分段回归、鲁棒建模和缺失数据处理等场景中给出可靠的解。

请帮我区分理解清楚潜变量、混合模型、EM以及EM用在其他模型上的关系

已思考 12s

下面把四个概念“潜变量、混合模型、EM、EM在其他模型上的应用”拆开讲清，并把它们的关系串成一张思维图。读完你能分得很清楚：**是什么 → 解决什么 → 怎么连起来用。**

1) 潜变量 (Latent variables) 是什么？

- **定义：**数据里看不见但对观测有影响的变量。记作 z 。每个样本通常有自己的 z_i 。
- **作用：**
 1. 解释观测变量之间的相关性（“共同原因/结构”）；
 2. 作为**瓶颈/压缩表示**，做无监督表示学习；
 3. 让复杂分布可分解、可建模。
- **和“参数”区别：**参数 θ 对所有样本共享；潜变量 z_i 随样本而变。

2) 混合模型 (Mixture models) 是什么？

- **定义：**最简单、最常用的一类潜变量模型：

$$p(x) = \sum_{k=1}^K \pi_k p_k(x | \theta_k), \quad z \in \{1, \dots, K\}$$

这里 z 表示“这条样本来自哪个分量/簇”。

- **直觉**: 总体分布是若干“子分布”的**加权相加** (凸组合)。
- **例子**: GMM (高斯混合)、伯努利混合、混合的专家 (MoE, 权重还依赖输入 x)。
- **用途**: 密度估计、异常检测、生成式分类器 $p(x | y)$ 、**软聚类** (用后验 $r_{ik} = p(z_i = k | x_i)$)。

结论: 混合模型 \subset 潜变量模型 (是其中离散 z 的特例)。

3) EM 算法 (Expectation–Maximization) 是什么?

- **出场场景**: 当模型里有潜变量/缺失数据时, 观测对数似然 $\sum_i \log \sum_{z_i} p(x_i, z_i | \theta)$ 非凸且不好直接最大化。
- **做法 (两步交替)**:
 - **E 步**: 在旧参数 $\theta^{(t)}$ 下, 求潜变量的**后验期望或期望充分统计量 (ESS)**, 本质是“用概率把看不见的部分补上”。
 - **M 步**: 在“补好的统计量”上对 θ 做极大 (或加先验做 MAP)。
- **保证**: 每次迭代不降 (提高) 观测对数似然的**下界** \rightarrow 通常单调上升到一个**局部极大**。
- **特点**: 常有**闭式更新**、实现简单; 但**非凸/多峰** \rightarrow 依赖初始化; ML 可能奇异, MAP 更稳。

结论: EM 是“学参数”的通用办法, 只要“有潜变量 \rightarrow 完整数据下的极大容易做”, 就适用。

4) EM 用在其他模型上 (除了混合模型)

EM 不只用于 GMM, 只要我们能把模型写成“含潜变量的完整数据似然容易”的形式 (必要时人造一个辅助潜变量), 就能用。典型对应如下:

模型	潜变量 z 是谁	E 步算什么	M 步更新什么	直觉/用途
GMM	簇指示 $z_i \in \{1..K\}$	责任度 $r_{ik} = p(z_i = k x_i)$	π_k, μ_k, Σ_k	软聚类/密度

模型	潜变量 z 是谁	E 步算什么	M 步更新什么	直觉/用途
伯努利/多项式混合	簇指示	同上	各簇的参数 (像素/词的概率)	二值/词袋数据
Mixture of Experts (MoE)	选哪个专家	责任度 (但权重依赖 x)	专家回归的加权最小二乘; gating 的软标签逻辑回归	分段回归/多态输出
层次 MoE	路径上的选择	各层责任度	各层 gating 与专家参数	复杂分区
Student t	尺度因子 $z_i \sim$ Gamma (把 t 写成高斯尺度混合)	$E[z_i], E[\log z_i]$	μ, Σ (闭式) , v (1D 数值/GEM/ECME)	抗离群点
Probit 回归	连续阈值变量 $z_i \sim N(w^\top x_i, 1)$	截断高斯的 $E[z_i]$	岭回归求 w	二分类的 EM
有隐结点的 DGM	各隐结点	族边缘的期望计数	CPT/参数 (+伪计数得 MAP)	一般图模型学习
MVN 缺失值	缺失的分量	条件高斯的 $E[x_h], E[x_h x_h^\top]$	μ, Σ (用 ESS)	带 NaN 的高斯学习/插补

关键心法：看不见 → 先用 E 步“概率填充”→ 用 M 步像“完整数据”那样学参数。有时为了用 EM，会引入辅助潜变量（Student t、Probit 就是这样）。

5) 彼此关系一图懂

```
flowchart LR
    LVM[LVM[潜变量模型LVM\n(含隐藏z)]] --> MIX[MIX[混合模型\n(GMM/伯努利混合...)]]
    LVM --> OTHER[OTHER[其他含z的模型\n(MoE/Student/Probit/DGM/HMM/缺失数据...)]]]
    EM[EM[EM算法\n\ne:算z的后验期望\n\m:更新\theta]] -->|用于学习\theta| MIX
    EM -->|用于学习\theta| OTHER
    AUX[AUX[辅助潜变量/重写]] -->|把困难模型变成\n“完整数据好优化”| OTHER
    MAP[MAP[先验/正则]] -->|在M步加入| EM
```

6) 易混点 & 选择建议

- **混合模型 ≠ EM**: 混合模型是一种模型；EM是一类优化算法。只是 GMM 与 EM 常常一起出现。

- **什么时候用 EM**: 模型有潜变量/缺失数据, 且完整数据下的极大容易 (常见是指数族) → 选 EM; 否则直接用梯度法更省事。
 - **ML vs MAP**: GMM 的 ML 有“方差塌缩”奇异; 常用 **MAP (Dirichlet+NIW) ** 保稳。
 - **均值 vs 众数 (逆问题)** : 多模态输出时用“众数”更合理 (MoE 的一个卖点) 。
 - **非凸/多峰**: 多次重启、K-means++ 初始化、退火/过松弛/变分或 MC 变体, 可提速或避坑。
-

一句话总结

- **潜变量**: 看不见的原因/表示;
- **混合模型**: 带离散潜变量的 LVM, 整体是多个子分布的加权和;
- **EM**: 在有潜变量/缺失数据时学参数的通用迭代法 (先“补”再“学”) ;
- **EM+其他模型**: 只要能让“完整数据似然好算”(必要时加辅助潜变量), EM 就能成为那类模型的学习引擎。

因为不知道有没有潜变量, 如何判断是混合模型模型呢, 还是说所有的模型都可以是混合模型

已思考 21s

不是所有模型都是混合模型。混合模型是一类很具体的潜变量模型: 在每个样本里新增一个离散的隐藏变量 $z_i \in \{1, \dots, K\}$ 来“选择”哪一个子分布生成该样本,

$$p(x) = \sum_{k=1}^K \pi_k p_k(x | \theta_k).$$

只有当你的建模假设或数据特征指向“由若干子群/机制拼在一起”时, 混合模型才合适。

下面给你一套判断与选择的“清单”。

1) 什么时候考虑“这是个混合模型”?

现象或先验知识表明存在多个子群或多种机制:

1. **多峰/多模态**: 边缘分布或某些投影出现不止一个峰 (例如身高的男女两峰) 。

2. **异质性**: 同一总体里混着几类来源/工艺/人群、不同噪声水平、不同形状。
3. **要做聚类**: 希望给样本分组且能给出“软归属概率”。
4. **条件多模态**: 给定 x 时 y 可能有多种可能 (逆问题、分段规律) → 用**混合的专家 (MoE) **或条件混合。
5. **形状复杂**: 整体密度不像单个高斯那样“一个椭球”，而是由多个“团块”组成——GMM 能把复杂形状拼出来。

小提醒：混合模型的“簇”**未必对应真实种类**，更像是统计上的“解释子”。解释要谨慎。

2) 什么时候不要上混合模型？

- 分布**单峰但尾部重**或有少量离群: 用 Student-t (鲁棒) 或稳健回归即可。
- 结构更像**连续因子**而非离散子群: 用 PCA/FA/ICA/**变分自编码器**等连续潜变量模型。
- 只需单一平滑映射的回归/分类: 用线性/核/树/NN; MoE 只有在**分段/多模态**输出时才必要。
- 你只想要通用密度估计而非可解释簇: 可用 KDE、**正态化流**、**树/森林密度**, 不一定要混合。

3) 实操判断步骤 (不纠结“真有没有潜变量”)

1. **画图看形状**: 直方图/KDE、pair plot。若明显多峰或团块 → 倾向混合。
2. **单分量基线**: 先拟合“一个分量”的模型 (如单高斯/单回归)，做留出或CV的负对数似然(NLL)。
3. **加混合对比**: 试 $K = 2, 3, \dots$, 比较BIC/验证NLL (概率模型) 或任务指标 (生成式分类、异常检测)。
4. **看责任度** $r_{ik} = p(z_i = k | x_i)$: 若大多接近 0/1, 说明确有可分子群; 若都“半半”, 混合的价值有限。
5. **可解释性**: 分量均值/协方差是否与业务子群对应? 若完全不可解释, 仅把它当密度近似器即可。
6. **稳健性**: GMM 选 MAP (Dirichlet+NIW 先验) ; 多次重启/K-means++ 初始化; 防“方差塌缩”。

4) “所有模型都能是混合吗？”

- **从逼近角度：**给足够多分量，**高斯混合能逼近任何连续密度**；形式上很多分布都可写成某种“混合”。
 - **从建模角度：**这不等于“都应该用混合”。混合引入**离散潜变量** → 优化非凸、需定 K 、易过拟合/奇异；且分量常难解释。
 - **结论：**混合是一种**可选假设/工具**，不是普适真相。若缺乏多机制证据、或目标不需要聚类/多模态，没必要强行用混合。
-

5) 快速决策表

- **看到多峰/多子群证据？** 是 → 试混合 (GMM/伯努利混合/MoE) ；否 → 用单模型或鲁棒/连续潜变量。
 - **任务=聚类/生成式分类/异常检测？** 是 → 混合常是好起点。
 - **固定 x 下 y 多解？** 是 → MoE 或条件混合；用众数而非均值做预测。
 - **单峰 + 重尾/离群？** → Student-t (或稳健版本) 。
 - **只要密度近似且不在乎解释** → 混合可用，但也评估 KDE/流模型的效果与成本。
-

一句话

混合模型 = 离散潜变量选择子机制的假设。当数据/领域知识支持“有多个子群或多模态”时用它；否则用单分量、鲁棒或连续潜变量模型往往更简单稳妥。