

翻译和解释文本

With the ever increasing amounts of data in electronic form, the need for automated methods for data analysis continues to grow. The goal of machine learning is to develop methods that can automatically detect patterns in data, and then to use the uncovered patterns to predict future data or other outcomes of interest. Machine learning is thus closely related to the fields of statistics and data mining, but differs slightly in terms of its emphasis and terminology. This book provides a detailed introduction to the field, and includes worked examples drawn from application domains such as molecular biology, text processing, computer vision, and robotics. This book is suitable for upper-level undergraduate students and beginning graduate students in computer science, statistics, electrical engineering, econometrics, or any one else who has the appropriate mathematical background. Specifically, the reader is assumed to already be familiar with basic multivariate calculus, probability, linear algebra, and computer programming. Prior exposure to statistics is helpful but not necessary.

This book adopts the view that the best way to make machines that can learn from data is to use the tools of probability theory, which has been the mainstay of statistics and engineering for centuries. Probability theory can be applied to any problem involving uncertainty. In machine learning, uncertainty comes in many forms: what is the best prediction (or decision) given some data? what is the best model given some data? what measurement should I perform next? etc. The systematic application of probabilistic reasoning to all inferential problems, including inferring parameters of statistical models, is sometimes called a Bayesian approach. However, this term tends to elicit very strong reactions (either positive or negative, depending on who you ask), so we prefer the more neutral term “probabilistic approach”. Besides, we will often use techniques such as maximum likelihood estimation, which are not Bayesian methods, but certainly fall within the probabilistic paradigm. Rather than describing a cookbook of different heuristic methods, this book stresses a principled model-based approach to

machine learning. For any given model, a variety of algorithms can often be applied. Conversely, any given algorithm can often be applied to a variety of models. This kind of modularity, where we distinguish model from algorithm, is good pedagogy and good engineering. We will often use the language of graphical models to specify our models in a concise and intuitive way. In addition to aiding comprehension, the graph structure aids in developing efficient algorithms, as we will see. However, this book is not primarily about graphical models; it is about probabilistic modeling in general.

Nearly all of the methods described in this book have been implemented in a MATLAB software package called PMTK, which stands for probabilistic modeling toolkit. This is freely available from pmtk3.googlecode.com (the digit 3 refers to the third edition of the toolkit, which is the one used in this version of the book). There are also a variety of supporting files, written by other people, available at pmtksupport.googlecode.com. These will be downloaded automatically, if you follow the setup instructions described on the PMTK website. MATLAB is a high-level, interactive scripting language ideally suited to numerical computation and data visualization, and can be purchased from www.mathworks.com. Some of the code requires the Statistics toolbox, which needs to be purchased separately. There is also a free version of Matlab called Octave, available at <http://www.gnu.org/software/octave/>, which supports most of the functionality of MATLAB. Some (but not all) of the code in this book also works in Octave. See the PMTK website for details. PMTK was used to generate many of the figures in this book; the source code for these figures is included on the PMTK website, allowing the reader to easily see the effects of changing the data or algorithm or parameter settings. The book refers to files by name, e.g., `naiveBayesFit`. In order to find the corresponding file, you can use two methods: within Matlab you can type `which naiveBayesFit` and it will return the full path to the file; or, if you do not have Matlab but want to read the source code anyway, you can use your favorite search engine, which should return the corresponding file from the pmtk3.googlecode.com website. Details on how to use PMTK can be found on the website, which will be updated

over time. Details on the underlying theory behind these methods can be found in this book.

以下是该段英文文本的中文翻译和解释：

中文翻译：

随着以电子形式存在的数据量不断增加，对自动化数据分析方法的需求也日益增长。机器学习的目标是开发能够自动识别数据中模式的方法，并利用这些已识别的模式来预测未来的数据或其他感兴趣的结果。因此，机器学习与统计学和数据挖掘密切相关，但在强调点和术语方面略有不同。

本书对该领域进行了详细的介绍，并包含了来自诸如分子生物学、文本处理、计算机视觉和机器人学等应用领域的实例。本书适用于高年级本科生和计算机科学、统计学、电气工程、计量经济学等专业的初级研究生，或者任何具备相应数学背景的读者。具体而言，假定读者已掌握基础的多元微积分、概率论、线性代数和计算机编程知识。虽然有统计学的背景会有所帮助，但不是必需的。

本书采取的观点是：从数据中学习的最佳方法是使用概率论工具，概率论在统计学和工程领域已经被使用了几个世纪。概率论适用于任何涉及不确定性的问题。在机器学习中，不确定性有多种形式，例如：给定数据，最好的预测或决策是什么？给定数据，最好的模型是什么？下一步应进行哪种测量？等等。

将概率推理系统性地应用于所有推断问题（包括推断统计模型的参数）的方法，有时被称为贝叶斯方法。但由于“贝叶斯”这个词往往引发强烈反应（正面或负面，取决于听者），因此本书更倾向于使用中性的术语“概率方法（probabilistic approach）”。此外，本书也会使用一些不属于贝叶斯方法但仍属于概率范式的技术，如最大似然估计（maximum likelihood estimation）。

与其列出各种启发式方法的“食谱”，本书更强调一种基于模型的原则性方法。对于任何给定的模型，往往可以应用多种算法；反过来，对于任何一种算法，也可以应用于多种模型。这种模型与算法相分离的模块化设计既有助于教学也利于工程实践。我们常常使用**图模型（graphical models）**的语言来简明直观地描述模型。图结构不仅有助于理解，也有助于开发高效算法——本书会对此进行说明。然而，本书的重点不是图模型本身，而是更广义的概率建模。

本书介绍的大多数方法都已经通过一个名为 PMTK（probabilistic modeling toolkit）的 MATLAB 软件工具包实现。该工具包可在 pmtk3.googlecode.com 上免费下载（“3”指的是该工具包的第三版，对应本书的版本）。还有一些由他人编写的辅助文件可在

pmtksupport.googlecode.com 下载；按照 PMTK 网站上的设置说明操作，这些文件会自动下载。

MATLAB 是一种高级交互式脚本语言，非常适合数值计算和数据可视化，可从 www.mathworks.com 购买。部分代码需要使用 MATLAB 的 Statistics Toolbox，该工具箱需另购。还有一个 MATLAB 的免费替代品叫 Octave，可在 <http://www.gnu.org/software/octave/> 下载，它支持 MATLAB 的大多数功能。本书中部分代码也支持 Octave（但并非全部）。详情请参考 PMTK 网站。

本书中许多图表都是使用 PMTK 生成的，其源代码可在 PMTK 网站上找到，读者可以通过修改数据、算法或参数设置轻松观察变化效果。书中会引用文件名，例如 `naiveBayesFit`，你可以通过以下两种方法找到对应的文件：

1. 在 MATLAB 中输入 `which naiveBayesFit`，它会返回该文件的完整路径；
2. 如果没有 MATLAB，但仍想查看源码，可以在搜索引擎中输入文件名，一般会返回该文件在 pmtk3.googlecode.com 上的链接。

关于 PMTK 的具体使用方法可参考其网站（会持续更新），而这些方法背后的理论依据则详见本书内容。

解释与重点：

- **核心主题：** 本书是一本关于概率方法的机器学习教材，强调以数学严谨的方式理解和构建模型。
- **适用人群：** 具备一定数学背景的本科高年级或研究生，特别是那些熟悉微积分、线性代数、概率论和编程的人。
- **方法论立场：**
 - 采用概率视角，而不是“黑盒”式的机器学习方法；
 - 支持但不完全局限于贝叶斯方法；
 - 强调模型与算法的分离（模块化设计思维）；
 - 强调图模型作为辅助工具，而非中心主题。
- **软件工具支持：** 本书配套的 MATLAB 工具包（PMTK）使读者可以亲自实验模型和算法，增进理解。
- **学习建议：** 如果你不熟悉 MATLAB，可以考虑用 Octave；动手运行源码、修改参数将有助于加深理解。

如果你想要深入学习该书的内容或需要相关章节逐步解析，也可以告诉我。

It should be noted, however, that even when one has an apparently massive data set, the effective number of data points for certain cases of interest might be quite small. In fact, data across a variety of domains exhibits a property known as the long tail, which means that a few things (e.g., words) are very common, but most things are quite rare (see Section 2.4.6 for details). For example, 20% of Google searches each day have never been seen before⁴. This means that the core statistical issues that we discuss in this book, concerning generalizing from relatively small samples sizes, are still very relevant even in the big data era.

以下是这段英文的中文翻译与解释：

中文翻译：

然而需要注意的是，即使我们拥有一个看起来非常庞大的数据集，在某些特定情境下的有效数据量仍然可能非常小。事实上，很多领域的数据都表现出一种被称为**“长尾现象”（long tail）“的特性，即少数事物（如常见词语）出现频率很高，但绝大多数事物非常罕见**（详见第2.4.6节）。

例如，谷歌每天有 20% 的搜索请求是前所未见的。这意味着，即使在大数据时代，我们在本书中讨论的关于如何从相对较小的样本中进行泛化（generalization）的问题，依然非常重要且相关的。

解释与重点：

- **长尾现象：**

在很多数据分布中（如词频、搜索请求、用户行为等），少数“热门”项占据了大量出现频次，而剩下的“大多数”项虽然各自频率低，但种类丰富，构成了“长尾”。这使得即便整体数据量庞大，每个罕见项的样本仍然很少，从统计角度看仍是“小数据”。

- **有效样本量不足：**

举个例子，假设你有一个含有10亿条记录的数据集，但你关注的是其中某个罕见现象，它可能只出现了几十次。这种情况下，对该现象建模时依然要面对样本稀缺的问题。

- **大数据 ≠ 无需泛化能力:**

大数据时代并没有让传统统计/机器学习的核心问题过时，比如：如何从小样本中推断整体趋势、如何避免过拟合、如何建模稀有事件等。

- **实际意义：**

这强调了在大数据应用中，不仅要处理数量庞大的“常见”情况，更要有能力应对**长尾中的少数样本或未知情境**——这正是概率模型与泛化能力发挥作用的地方。

如果你还想更深入地理解“长尾分布”、“泛化能力”或这段内容背后的数学基础，也可以继续问我。

Machine learning is usually divided into two main types. In the predictive or supervised learning approach, the goal is to learn a mapping from inputs x to outputs y , given a labeled set of input-output pairs $D = \{(x_i, y_i)\}_{i=1}^N$. Here D is called the training set, and N is the number of training examples. In the simplest setting, each training input x_i is a D -dimensional vector of numbers, representing, say, the height and weight of a person. These are called features, attributes or covariates. In general, however, x_i could be a complex structured object, such as an image, a sentence, an email message, a time series, a molecular shape, a graph, etc. Similarly the form of the output or response variable can in principle be anything, but most methods assume that y_i is a categorical or nominal variable from some finite set, $y_i \in \{1, \dots, C\}$ (such as male or female), or that y_i is a real-valued scalar (such as income level). When y_i is categorical, the problem is known as classification or pattern recognition, and when y_i is real-valued, the problem is known as regression. Another variant, known as ordinal regression, occurs where label space Y has some natural ordering, such as grades A–F.

The second main type of machine learning is the descriptive or unsupervised learning approach. Here we are only given inputs, $D = \{x_i\}_{i=1}^N$, and the goal is to find “interesting patterns” in the data. This is sometimes called knowledge discovery. This is a much less well-defined problem, since we are not told what kinds of patterns to

look for, and there is no obvious error metric to use (unlike supervised learning, where we can compare our prediction of y for a given x to the observed value). There is a third type of machine learning, known as reinforcement learning, which is somewhat less commonly used. This is useful for learning how to act or behave when given occasional reward or punishment signals. (For example, consider how a baby learns to walk.) Unfortunately, RL is beyond the scope of this book, although we do discuss decision theory in Section 5.7, which is the basis of RL. See e.g., (Kaelbling et al. 1996; Sutton and Barto 1998; Russell and Norvig 2010; Szepesvari 2010; Wiering and van Otterlo 2012) for more information on RL.

以下是这段英文内容的**中文翻译与解释**:

中文翻译:

机器学习通常分为两大类：

1. 预测性（监督式）学习（Predictive / Supervised Learning）

在这种方法中，目标是：**学习一个从输入 x 到输出 y 的映射关系**，前提是拥有标注好的输入-输出对：

$$D = \{(x_i, y_i)\}_{i=1}^N$$

其中， D 被称为**训练集**（training set）， N 是训练样本的数量。

在最简单的情况下，每个训练输入 x_i 是一个 D 维的数值向量，例如表示一个人的身高和体重。这些输入特征也被称为**特征**（features）、**属性**（attributes）或**协变量**（covariates）。

但在更一般的情形下，输入 x_i 可以是更复杂的结构化对象，如：一张图片、一段句子、一封电子邮件、一条时间序列、一个分子形状、一个图结构等。

输出 y_i 的形式也可以是多种多样的，但多数方法假设：

- **分类问题**（Classification）： y_i 是一个**类别变量**，来自某个有限集合（如：性别是“男”或“女”）。
- **回归问题**（Regression）： y_i 是一个**实数值**（如：收入水平）。

还有一种变体叫做**有序回归 (Ordinal Regression)**，适用于输出有自然顺序的情况，比如成绩等级 A–F。

2. 描述性 (无监督) 学习 (Descriptive / Unsupervised Learning)

在这种方法中，只有输入数据：

$$D = \{x_i\}_{i=1}^N$$

目标是：**在数据中发现“有趣的模式”**，这也常被称为**知识发现 (knowledge discovery)**。

与监督学习不同，无监督学习的问题定义更模糊：

- 没有明确告诉你要找什么样的模式；
 - 也没有明显的“错误度量标准” (error metric) 来判断结果对错。
-

3. 强化学习 (Reinforcement Learning, 简称 RL)

这是第三种机器学习形式，相对较少使用。其目标是：**在面对奖励和惩罚信号的情况下学习如何行动或做决策**。例如，婴儿学习走路的过程就类似于强化学习。

本书**不深入讨论强化学习**，但会在第 5.7 节讨论其基础：**决策理论 (decision theory)**。

若读者对 RL 感兴趣，可参考以下资料：

- Kaelbling et al. (1996)
 - Sutton and Barto (1998)
 - Russell and Norvig (2010)
 - Szepesvari (2010)
 - Wiering and van Otterlo (2012)
-

解释与重点总结：

类别	输入	输出	目标	示例	是否有“老师”
监督学习	有	有	学习输入到输出的映射	图像识别、房价预测	<input checked="" type="checkbox"/> (有标签)
无监督学习	有	无	找出隐藏模式或结构	聚类、主题建模	<input type="checkbox"/> (无标签)

类别	输入	输出	目标	示例	是否有“老师”
强化学习	状态	奖励信号	学习最优行为策略	机器人控制、游戏AI	<input checked="" type="checkbox"/> (但反馈是延迟的)

- **监督学习** = 有目标指导的学习。
- **无监督学习** = 自主探索数据结构。
- **强化学习** = 通过“试错”学会行为。

如果你希望我进一步详细解释某种类型的机器学习，或者举一些具体例子（如图像识别、客户分群、股市预测等），可以随时告诉我！

Figure 1.1 Left: Some labeled training examples of colored shapes, along with 3 unlabeled test cases. Right: Representing the training data as an $N \times D$ design matrix. Row i represents the feature vector x_i . The last column is the label, $y_i \in \{0, 1\}$. Based on a figure by Leslie Kaelbling.

以下是该段描述的中文翻译与解释：

中文翻译：

图 1.1

左图：一些带标签的训练样本，包括不同颜色的几何形状，以及三个未标记的测试样本。

右图：将训练数据表示为一个 $N \times D$ 的**设计矩阵** (design matrix)。

矩阵中的第 i 行表示特征向量 x_i ，最后一列是标签 $y_i \in \{0, 1\}$ 。

该图基于 Leslie Kaelbling 的图示绘制。

解释与补充说明：

1. 图的含义（文字描述）：

- **左侧图示：**

- 你会看到几个颜色和形状各异的图形（比如红色圆形、绿色三角形等），每个带有一个标签（如 0 或 1），表示其分类。

- 同时，还有 3 个没有标签的图形（即测试样本），系统的目标是预测它们属于哪一类（0 还是 1）。
- 右侧图示：**
 - 这是一个表格或矩阵，称为**设计矩阵**。
 - 其中：
 - 每行是一个训练样本的**特征向量** \mathbf{x}_i ；
 - 最后一列是该样本对应的**输出标签** y_i ，取值为 0 或 1；
 - 如果图形被用作输入，则特征可能表示“颜色编码”、“形状类型”等。

2. 矩阵结构：

一个典型的设计矩阵可能如下所示（仅为示意）：

颜色编码	形状编码	标签
1	0	0
0	1	1
1	1	0
...

- N ：训练样本数（矩阵的行数）；
- D ：每个样本的特征维度（列数，不包括标签）；
- 特征可以是数值、编码过的类别（如 one-hot），甚至是嵌入向量。

3. 背后思想：

- 可视化学习任务：**图 1.1 左图帮助你形象理解什么是“已知标签的训练数据”与“待预测的测试数据”；
- 数学建模方式：**右图展示了如何把这些样本转化为机器学习算法可以处理的结构化数据（矩阵形式）；
- 这是一种典型的监督学习问题**，因为我们有输入 \mathbf{x}_i 和对应的输出 y_i 。

如你希望我为这个图例补充绘图、具体数值示例或如何在 Python / MATLAB 中构造这样的设计矩阵，也可以告诉我！

In this section, we discuss classification. Here the goal is to learn a mapping from inputs x to outputs y , where $y \in \{1, \dots, C\}$, with C being the number of classes. If $C = 2$, this is called binary classification (in which case we often assume $y \in \{0, 1\}$); if $C > 2$, this is called multiclass classification. If the class labels are not mutually exclusive (e.g., somebody may be classified as tall and strong), we call it multi-label classification, but this is best viewed as predicting multiple related binary class labels (a so-called multiple output model). When we use the term “classification”, we will mean multiclass classification with a single output, unless we state otherwise. One way to formalize the problem is as function approximation. We assume $y = f(x)$ for some unknown function f , and the goal of learning is to estimate the function f given a labeled training set, and then to make predictions using $\hat{y} = \hat{f}(x)$. (We use the hat symbol to denote an estimate.) Our main goal is to make predictions on novel inputs, meaning ones that we have not seen before (this is called generalization), since predicting the response on the training set is easy (we can just look up the answer).

以下是这段英文内容的**中文翻译与解释**:

中文翻译:

在本节中，我们讨论**分类问题** (classification)。

这里的目标是：学习一个从输入 x 到输出 y 的映射，其中

$$y \in \{1, \dots, C\}$$

表示类别编号， C 是类别总数。

- 如果 $C = 2$ ，这称为**二元分类** (binary classification)，此时我们通常假设 $y \in \{0, 1\}$ ；
- 如果 $C > 2$ ，这称为**多类别分类** (multiclass classification)；
- 如果类别标签**不是互斥的**（例如某人可以同时被归类为“高个”和“强壮”），这被称为**多标签分类** (multi-label classification)。不过，更恰当的做法是把它看作在同时预测多个相关的二元标签，也就是所谓的**多输出模型** (multiple output model)。

在本书中，除非特别说明，我们使用“分类”一词时，**默认指的是“单输出的多类别分类”。**

分类问题可以形式化为一个**函数逼近问题**：

我们假设存在某个未知的函数 f ，使得

$$y = f(x)$$

学习的目标就是在给定一组带标签的训练数据的前提下，估计出这个函数 f ，并用它来进行预测：

$$\hat{y} = \hat{f}(x)$$

(我们使用“^”符号来表示估计值。)

机器学习的主要目标是在新样本（未见过的输入）上做出准确预测，这叫做**泛化**（generalization）。

在训练集上预测结果很容易（你可以直接“查答案”），但这并不能体现模型的真正能力。

解释与重点总结：

分类的三种形式：

类型	类别数	是否互斥	示例
二元分类	2	是	是狗 / 不是狗
多类别分类	>2	是	猫 / 狗 / 鸟
多标签分类	≥ 2	否	高个 & 强壮，可以共存

⚠ 注意：多标签分类 \neq 多类别分类。前者可以多个标签共存，后者标签之间互斥。

函数逼近视角：

- 把分类任务看成在估计一个函数 f ，这个函数接收输入 x ，输出对应的类标签 y 。
- 我们通过数据学习得到这个函数的近似 \hat{f} ，再用它来预测新数据的输出 \hat{y} 。
- 模型的真正挑战是能否**泛化到未见样本**，而不是记住训练集。

常见分类模型示例：

- 二元分类：逻辑回归、支持向量机（SVM）、朴素贝叶斯；
- 多类别分类：softmax 回归、多类决策树、KNN、多类 SVM；
- 多标签分类：二元分类器组合（one-vs-rest）、神经网络多输出层。

如果你希望我进一步解释分类算法、举图示例、展示代码实现（如在 Python 中用 scikit-learn 训练一个分类模型），也可以继续告诉我。

As a simple toy example of classification, consider the problem illustrated in Figure 1.1(a). We have two classes of object which correspond to labels 0 and 1. The inputs are colored shapes. These have been described by a set of D features or attributes, which are stored in an $N \times D$ design matrix X , shown in Figure 1.1(b). The input features x can be discrete, continuous or a combination of the two. In addition to the inputs, we have a vector of training labels y .

In Figure 1.1, the test cases are a blue crescent, a yellow circle and a blue arrow. None of these have been seen before. Thus we are required to generalize beyond the training set. A reasonable guess is that blue crescent should be $y = 1$, since all blue shapes are labeled 1 in the training set. The yellow circle is harder to classify, since some yellow things are labeled $y = 1$ and some are labeled $y = 0$, and some circles are labeled $y = 1$ and some $y = 0$. Consequently it is not clear what the right label should be in the case of the yellow circle. Similarly, the correct label for the blue arrow is unclear.

以下是这段内容的**中文翻译与解释**:

中文翻译:

作为一个简单的分类玩具例子，请参考图 1.1(a) 所示的问题。我们有两类对象，分别对应标签 0 和 1。输入是带颜色的图形（如形状和颜色的组合）。这些输入由一组 D 个**特征 (features)** 或**属性 (attributes)** 描述，并被存储在一个 $N \times D$ 的**设计矩阵 (design matrix)** X 中，如图 1.1(b) 所示。

输入特征 x 可以是**离散的 (如颜色)**、**连续的 (如大小)**，或二者的组合。除了输入以外，我们还拥有一个**训练标签向量 y** 。

在图 1.1 中，测试样本包括：

- 一个蓝色的新月形 (blue crescent) 、
- 一个黄色圆形 (yellow circle) 、
- 以及一个蓝色箭头 (blue arrow) 。

这些图形在训练集中都**从未出现过**，因此我们需要**对训练集以外的数据进行泛化 (generalization)**。

对于蓝色新月形，我们可以合理猜测其标签为 $y = 1$ ，因为训练集中所有蓝色的图形都被标记为 1。

但对于黄色圆形，分类就比较困难了，因为：

- 有些黄色的图形被标记为 $y = 1$ ，有些则是 $y = 0$ ；
- 有些圆形是 $y = 1$ ，有些也是 $y = 0$ 。

因此，对于黄色圆形来说，**没有明显的规律可以依赖**，很难判断它属于哪一类。同样，蓝色箭头的标签也不清晰。

解释与重点说明：

1. 这是一个典型的分类问题示例：

- 输入是图形 (形状 + 颜色) ；
- 输出是二分类标签 (0 或 1) ；
- 目标是从训练数据中学习规律，以便预测**新样本**的标签。

2. 设计矩阵 X ：

- 每行是一个样本；
- 每列是一个特征 (例如颜色编码、形状编码) ；
- 通常还会有对应的标签向量 y 。

3. 泛化能力的考验：

- 对于**蓝色新月形**：虽然没见过“新月”这种形状，但**颜色是蓝色**，而训练集中所有蓝色图形都是 $1 \rightarrow$ 有合理的泛化依据 \rightarrow 推测 $y = 1$ 。
- 对于**黄色圆形**：颜色和形状都在训练集中出现过，但各自对应的标签不一致 \rightarrow 模型难以判断。
- 对于**蓝色箭头**：颜色为蓝色 (可能提示标签为 1)，但箭头形状可能是新出现的 \rightarrow 模型泛化能力再次受考验。

4. 模型角度看：

- 若用的是决策树，可能会优先用颜色来分裂；
 - 若用的是逻辑回归，则会对每个特征赋权重；
 - 若是最近邻方法（KNN），预测结果可能取决于与其最接近的样本类别。
-

如果你希望我基于这个例子展示具体的编码实现（比如用 `scikit-learn` 建模并预测蓝色新月形的类别），或者可视化设计矩阵的样式，也欢迎继续提出！



这张图是对前面所提到的**图 1.1 的可视化示例**，它由两个部分组成：图 (a) 和图 (b)。我们逐一解释：

图 (a): 分类任务示意图

- 左边框标注为“yes”：表示这些图形属于类别 1（正类），对应标签为 1。
- 右边框标注为“no”：表示这些图形属于类别 0（负类），对应标签为 0。

下面有三个未标记的测试图形（蓝色新月形、黄色圆形、蓝色箭头），**每个后面都有问号**（?），表示我们需要预测它们的类别标签。

观察分析：

1. **蓝色新月形**：所有蓝色的图形在训练集中都在“yes”类，因此合理猜测它的标签是 1。
 2. **黄色圆形**：黄色的图形在“yes”和“no”类中都有，圆形也是 → 分类困难，不确定。
 3. **蓝色箭头**：蓝色暗示可能属于“yes”，但“箭头”在“no”类中出现过（红色箭头），→ 不确定。
-

图 (b): 设计矩阵 (Design Matrix) 与标签向量

这是将训练数据**结构化**为机器学习模型可以处理的形式：

- 表格的每一行表示一个样本（共 $N = 3$ 个样本）；
- 每列是一个特征，共有 $D = 3$ 个特征（颜色、形状、尺寸）；
- 右边是对应的标签向量 $y \in \{0, 1\}$ 。

Color	Shape	Size (cm)	Label
Blue	Square	10	1
Red	Ellipse	2.4	1
Red	Ellipse	20.7	0

说明：

- 第一个样本是蓝色方形，尺寸 10cm，属于类别 1；
- 第二个是红色椭圆，尺寸 2.4cm，属于类别 1；
- 第三个也是红色椭圆，但尺寸 20.7cm，属于类别 0。

模型训练的目标：

基于这三条样本，学习一个函数：

$$f(x) \approx y$$

使得对于新的图形（如蓝色新月形），可以估计它的标签 $\hat{y} = f(x)$ 。

✓ 总结：

- 图 (a)：是问题的直观描述（分类任务）；
- 图 (b)：是机器学习中对输入数据的**数值/结构化表示**；
- 此示例展示了：即使是图像/图形识别问题，也可以通过提取颜色、形状、大小等特征，转换成表格数据用于建模；
- 泛化能力很关键，尤其是面对**测试集中未见过的组合**时。

如你希望我基于这组数据构建一个实际的分类模型（例如用 Python 的 scikit-learn 实现），或将其转换为可输入模型的数值编码形式，我可以为你演示。是否需要继续？

To handle ambiguous cases, such as the yellow circle above, it is desirable to return a probability. The reader is assumed to already have some familiarity with basic concepts in probability. If not, please consult Chapter 2 for a refresher, if necessary.

We will denote the probability distribution over possible labels, given the input vector x and training set D by $p(y|x, D)$. In general, this represents a vector of length C . (If there are just two classes, it is sufficient to return the single number $p(y=1|x, D)$, since $p(y=1|x, D) + p(y=0|x, D) = 1$.) In our notation, we make explicit that the probability is conditional on the test input x , as well as the training set D , by putting these terms on the right hand side of the conditioning bar $|$. We are also implicitly conditioning on the form of model that we use to make predictions. When choosing between different models, we will make this assumption explicit by writing $p(y|x, D, M)$, where M denotes the model. However, if the model is clear from context, we will drop M from our notation for brevity.

Given a probabilistic output, we can always compute our “best guess” as to the “true label” using

$$\hat{y} = \hat{f}(x) = \underset{c=1}{\operatorname{argmax}} p(y=c|x, D) \quad (1.1)$$

This corresponds to the most probable class label, and is called the mode of the distribution $p(y|x, D)$; it is also known as a MAP estimate (MAP stands for maximum a posteriori). Using the most probable label makes intuitive sense, but we will give a more formal justification for this procedure in Section 5.7.

Now consider a case such as the yellow circle, where $p(\hat{y}|x, D)$ is far from 1.0. In such a case we are not very confident of our answer, so it might be better to say “I don’t know” instead of returning an answer that we don’t really trust. This is particularly important in domains such as medicine and finance where we may be risk averse, as we explain in Section 5.7. Another application where it is important to assess risk is when playing TV game shows, such as Jeopardy. In this game, contestants have to solve various word puzzles and answer a variety of trivia questions, but if they answer incorrectly, they lose money. In 2011, IBM unveiled a computer system called Watson

which beat the top human Jeopardy champion. Watson uses a variety of interesting techniques (Ferrucci et al. 2010), but the most pertinent one for our present purposes is that it contains a module that estimates how confident it is of its answer. The system only chooses to “buzz in” its answer if sufficiently confident it is correct. Similarly, Google has a system known as SmartASS (ad selection system) that predicts the probability you will click on an ad based on your search history and other user and ad-specific features (Metz 2010). This probability is known as the click-through rate or CTR, and can be used to maximize expected profit. We will discuss some of the basic principles behind systems such as SmartASS later in this book.

以下是这段内容的**中文翻译与解释**:

中文翻译:

为了应对像**黄色圆形**这类**分类结果不确定的情况**, 我们希望模型不仅输出一个分类结果, 还能给出一个**概率估计**。

我们假设读者已经具备**概率论的基础知识**; 如果不熟悉, 可参考第 2 章的内容进行复习。

我们将**给定输入向量 x 和训练集 D** 的情况下, 标签的概率分布表示为:

$$p(y|x, D)$$

这通常表示一个长度为 C 的向量 (C 是类别数)。

- 如果只有两个类别, 我们只需要返回一个数值, 比如:

$$p(y = 1|x, D)$$

因为:

$$p(y = 1|x, D) + p(y = 0|x, D) = 1$$

在本书的记法中, 我们明确表示概率是在**输入 x 和训练集 D** 给定的条件下计算的。另外, 虽然没有写出来, 但我们实际上还在**条件概率中隐含地依赖于所用的模型形式**。

如果我们需要比较不同模型的表现, 我们就会显式写出模型 M , 即:

$$p(y|x, D, M)$$

但如果上下文清楚，我们就会省略 M 以简化符号。

如何做出预测？

给定一个概率分布，我们可以使用下式输出最可能的类别（即“最佳猜测”）：

$$\hat{y} = \hat{f}(x) = \arg \max_{c=1, \dots, C} p(y = c | x, D) \quad (1.1)$$

这个预测值对应于概率分布 $p(y|x, D)$ 的**众数 (mode)**，也称为**最大后验估计 (MAP, Maximum A Posteriori)**。

用概率最高的类别作为预测是直观上合理的做法，我们将在**第 5.7 节**对此进行更严格的理论解释。

当模型不确定时该怎么办？

以**黄色圆形**为例，如果：

$$p(\hat{y}|x, D) \ll 1.0$$

说明模型对这个预测不太自信。此时，与其给出一个低可信度的答案，不如直接说“我不知道”。

在**高风险领域**（如**医疗或金融**）中尤其如此——我们宁愿不做决策，也不冒着做错的风险。第 5.7 节中将进一步讨论这一点。

现实中的应用案例：

1. IBM Watson (问答系统, 2011) :

- 在电视问答节目《Jeopardy》中击败人类冠军；
- 该系统内置一个模块，用于估计其答案的**置信度 (confidence)**；
- 只有在“足够自信”时才会“抢答” (buzz in)。

2. Google SmartASS (广告推荐系统) :

- 预测用户点击广告的概率（称为**点击率 CTR, Click-Through Rate**）；
 - 输入包括用户搜索历史、用户特征、广告特征等；
 - 系统基于 CTR 估计来**最大化广告收益**。
-

总结重点：

概念	说明
($p(y x, D)$)	
($\hat{y} = \arg\max p(y x, D)$)	
不确定预测	如果最大概率很低，应考虑拒绝预测（比如说“我不知道”）
应用场景	医疗诊断、金融风险控制、问答系统、广告系统等
点击率 (CTR)	用户点击广告的概率，用于收益最大化

如果你想深入了解如何在模型中实现**概率输出**、****阈值拒绝策略**（例如“不确定就不预测”）或如何用 Python 构建类似的模型，我可以帮你做个完整演示。是否需要继续？

Figure 1.2 Subset of size 16242 x 100 of the 20-newsgroups data. We only show 1000 rows, for clarity. Each row is a document (represented as a bag-of-words bit vector), each column is a word. The red lines separate the 4 classes, which are (in descending order) comp, rec, sci, talk (these are the titles of USENET groups). We can see that there are subsets of words whose presence or absence is indicative of the class. The data is available from <http://cs.nyu.edu/~roweis/data.html>. Figure generated by newsgroupsVisualize.

以下是该段内容的**中文翻译与解释**：

中文翻译：

图 1.2：

这是 20-Newsgroups 数据集中一个大小为 16242×100 的子集。为方便展示，仅显示了其中的 1000 行。

- 每一行代表一篇文档，以“词袋模型 (bag-of-words) ”形式表示成一个**比特向量**；

- 每一列代表一个单词；
- 红色的分隔线将数据分为 4 个类别，分别是（按顺序）：
 1. comp (计算机类)
 2. rec (娱乐/体育类)
 3. sci (科学类)
 4. talk (政治/宗教讨论类)

这些类别对应的是 USENET 论坛上的主题板块。

从图中可以看到：

在某些类别中，**某些单词的出现或缺失**具有很强的**类别指示性**（indicative of the class）。

该数据集可从此网址获取：

<http://cs.nyu.edu/~roweis/data.html>

图像由 `newsgroupsVisualize` 工具生成。

解释与重点说明：

什么是 20-Newsgroups 数据集？

- 一个经典的文本分类数据集；
- 包含来自 20 个不同 USENET 新闻组（网络讨论版）的文本帖子；
- 常用于测试**文本分类、主题建模、文档聚类等自然语言处理任务**。

图像结构说明：

- 图是一个稀疏矩阵的可视化：
 - 行 = 文档（1000 个）；
 - 列 = 单词（100 个）；
 - 每个点表示这个文档是否包含这个单词（1 表示出现，0 表示未出现）；
- 红线将不同类别（主题）分隔开，方便对比；
- 可视化直观显示了某些词只出现在特定类型的文章中。

词袋模型（Bag-of-Words, BoW）：

- 把每篇文档表示成一个不考虑词序的**词出现集合**；
- 每个文档被转换成一个固定长度的向量，每个位置对应一个词是否出现；
- 这种模型**简单但高效**，广泛用于文本分类。

🎯 分类的意义：

- 不同类别的文档使用的词语分布不同；
 - 例如，“comp”类中可能频繁出现 “computer”、“file”、“program”；
 - 而“rec”类可能更多出现 “game”、“team”、“score” 等；
 - 这说明了文本的“语言特征”可以强烈暗示文档主题。
-

✓ 总结：

要素	含义
行数	文档数 (这里只展示了 1000 行)
列数	单词数 (子集中选了 100 个)
值类型	0 或 1，表示某单词是否在该文档中出现
红线	不同类别的分隔线，共 4 类 (comp, rec, sci, talk)
数据表示	词袋模型，适用于分类和聚类
工具	<code>newsgroupsVisualize</code> 用于生成图像

如果你需要我用这类数据构建一个**文本分类模型示例**（如用朴素贝叶斯、逻辑回归等），或者讲解如何用 Python 处理这类稀疏矩阵（如用 `scikit-learn` 加载与可视化），我可以带你一步步做！需要吗？

In document classification, the goal is to classify a document, such as a web page or email message, into one of C classes, that is, to compute $p(y = c|x, D)$, where x is some representation of the text. A special case of this is email spam filtering, where the classes are spam $y = 1$ or ham $y = 0$.

Most classifiers assume that the input vector x has a fixed size. A common way to represent variable-length documents in feature-vector format is to use a bag of words representation. This is explained in detail in Section

3.4.4.1, but the basic idea is to define $x_{ij} = 1$ iff word j occurs in document i . If we apply this transformation to every document in our data set, we get a binary document \times word co-occurrence matrix: see Figure 1.2 for an example. Essentially the document classification problem has been reduced to one that looks for subtle changes in the pattern of bits. For example, we may notice that most spam messages have a high probability of containing the words "buy", "cheap", "viagra", etc. In Exercise 8.1 and Exercise 8.2, you will get hands-on experience applying various classification techniques to the spam filtering problem.

以下是这段内容的**中文翻译与解释**:

中文翻译:

在**文档分类 (document classification) **任务中，我们的目标是将一个文档（例如网页或电子邮件）分类到 C 个类别中的某一个，即计算：

$$p(y = c \mid x, D)$$

其中：

- x 是对文本的一种表示方式；
- D 是训练集；
- y 是文档所属的类别。

一个典型的特殊情况是：**电子邮件垃圾过滤 (email spam filtering)**，其中类别为：

- 垃圾邮件： $y = 1$
 - 正常邮件（也称为 "ham"）： $y = 0$
-

大多数分类器都假设**输入向量 x 的维度是固定的**。然而，文本文档长度各不相同，因此我们需要将变长的文本转换为固定维度的特征向量。

一个常用方法是采用**词袋模型 (bag-of-words representation)**。这个方法在第 3.4.4.1 节有详细解释，但其基本思想是：

$$x_{ij} = 1 \quad \text{当且仅当第 } j \text{ 个词在第 } i \text{ 篇文档中出现}$$

如果我们将对数据集中的每一篇文档都应用这种转换，就会得到一个**二值文档 × 单词的共现矩阵**，如图 1.2 所示。

本质上，这将文档分类问题转化为一个**在比特模式中识别微妙变化**的问题。

例如，我们可能注意到，垃圾邮件中往往更可能包含词汇如：

- “buy” (购买)
 - “cheap” (便宜)
 - “viagra” (伟哥)
- 等等。

你将在**习题 8.1 和 8.2**中亲自实践，学习如何将各种分类技术应用于垃圾邮件过滤任务。

解释与重点说明：

文档分类问题目标：

- 给定文档 x ，预测其所属的类别 $y \in \{1, \dots, C\}$ ；
- 计算的是条件概率 $p(y = c | x, D)$ ；
- 常见任务包括新闻分类、邮件过滤、网页内容分组等。

词袋模型 (Bag-of-Words, BoW) 思想：

- **忽略词序和语法结构**，只考虑某词是否出现在文档中；
- 每个文档被表示为一个向量，维度等于词汇表大小；
- 向量的每个位置代表一个单词是否出现（1 或 0）；
- 简单高效，便于转为机器学习可处理的特征格式。

12 二值文档-词矩阵 (如图 1.2) :

- 行表示文档；
- 列表示词汇；
- 值为 1 表示该单词在该文档中出现，0 表示未出现。

垃圾邮件分类作为案例：

- 类别只有两个：Spam (垃圾) 和 Ham (正常) → 二分类问题；
- 分类器根据某些关键词的出现（如 "viagra", "free", "discount"）学习如何判断邮件是否为垃圾。

后续实践内容：

- 第 8 章的习题将通过动手操作，使用真实数据和算法（如朴素贝叶斯、逻辑回归、支持向量机等）训练分类器；
 - 你将深入理解如何将文本转为特征、训练模型并评估性能。
-

总结表：

项目	说明
任务	文档分类：预测文档属于哪个类别
特例	垃圾邮件分类 (spam vs. ham)
特征表示	词袋模型：文档 → 二进制词向量
关键工具	文档-词共现矩阵 (如图 1.2 所示)
学习目标	学会处理文本输入，训练分类模型
实践入口	习题 8.1 和 8.2 (动手构建垃圾邮件过滤器)

如果你想继续，我可以帮你：

- 把一个简单文本数据集转为 BoW 表示；
- 使用 Python 建立垃圾邮件分类模型；
- 可视化文档-词矩阵；
- 或解释“如何从 $p(y | x, D)$ 得到分类决策”。

告诉我你的兴趣点即可！

Figure 1.3 Three types of iris flowers: setosa, versicolor and virginica. Source: <http://www.statlab.uni-heidelberg.de/data/iris/>. Used with kind permission of Dennis Krumb and SIGNA.

Figure 1.4 Visualization of the Iris data as a pairwise scatter plot. The diagonal plots the marginal histograms of the 4 features. The off diagonals contain scatterplots of all possible pairs of features. Red circle = setosa, green

diamond = versicolor, blue star = virginica. Figure generated by fisheririsDemo.

Figure 1.3 gives another example of classification, due to the statistician Ronald Fisher. The goal is to learn to distinguish three different kinds of iris flower, called setosa, versicolor and virginica. Fortunately, rather than working directly with images, a botanist has already extracted 4 useful features or characteristics: sepal length and width, and petal length and width. (Such feature extraction is an important, but difficult, task. Most machine learning methods use features chosen by some human. Later we will discuss some methods that can learn good features from the data.) If we make a scatter plot of the iris data, as in Figure 1.4, we see that it is easy to distinguish setosas (red circles) from the other two classes by just checking if their petal length or width is below some threshold. However, distinguishing versicolor from virginica is slightly harder; any decision will need to be based on at least two features. (It is always a good idea to perform exploratory data analysis, such as plotting the data, before applying a machine learning method.)

以下是这段内容的**中文翻译与详细解释**, 包括图 1.3 和图 1.4 的解读:

中文翻译:

图 1.3 展示了三种鸢尾花 (Iris flowers) 的图片:

- Setosa (山鸢尾)
- Versicolor (变色鸢尾)
- Virginica (维吉尼亚鸢尾)

来源: <http://www.statlab.uni-heidelberg.de/data/iris/>, 图像经 Dennis Kramb 和 SIGNA 授权使用。

图 1.4 是鸢尾花数据集的可视化, 采用**成对散点图 (pairwise scatter plot) **方式:

- 对角线显示了四个特征的边缘直方图;
- 非对角线区域显示了任意两个特征组合的散点图;
- 三个类别使用不同的标记区分:

- 红色圆点 = Setosa
- 绿色菱形 = Versicolor
- 蓝色星星 = Virginica

图像由 `fisheririsDemo` 生成。

正文翻译与说明：

图 1.3 是另一个分类问题的示例，由统计学家Ronald Fisher 提出。

任务目标是：学会区分三种鸢尾花 (Setosa、Versicolor 和 Virginica) 。

幸运的是，我们不需要直接处理图像数据，因为植物学家已经提取了 4 个有用的**特征 (features)**：

1. 萼片长度 (sepal length)
2. 萼片宽度 (sepal width)
3. 花瓣长度 (petal length)
4. 花瓣宽度 (petal width)

(注：**特征提取**是机器学习中一项非常重要但具有挑战性的任务。多数机器学习方法仍依赖人类选择好的特征。稍后我们将介绍一些**自动学习特征**的方法。)

如果我们像图 1.4 那样画出鸢尾花的散点图，就会发现：

- 要区分 Setosa (**红圈**) 和另外两类非常简单：
 - 只要检查花瓣的长度或宽度是否小于某个阈值即可；
- 但要区分 Versicolor (**绿菱形**) 和 Virginica (**蓝星号**) 就稍微困难些：
 - 通常需要至少**两个特征**来做出准确判断。

经验建议：

在应用机器学习方法之前，最好先进行**探索性数据分析 (EDA)**，例如画出可视化图表，这能帮助你了解数据的结构和分布特征。

✓ 总结与重点解读：

元素	内容
数据集	Fisher 的鸢尾花数据集 (Iris dataset) , 包含 150 条样本, 每类 50 条
特征数	4 个连续变量 (sepal 和 petal 的长度和宽度)
类别数	3 类 (Setosa、Versicolor、Virginica)
可视化技巧	Pairwise scatter plots + histogram
可区分性	Setosa 易于区分; Versicolor 和 Virginica 较难区分
实践建议	应先可视化再建模, 有助于了解数据结构
工具/图生成	使用 <code>fisheririsDemo</code> 绘制图 1.4

延伸说明：为何 scatter plot 有帮助？

- 你可以在图 1.4 中观察哪些特征组合对分类最有帮助；
- 比如花瓣长度 vs. 花瓣宽度图中, Setosa 类与其他类形成明显分离；
- 这可以帮助你选择合适的特征用于分类器训练。

如果你想我展示用 Python (如 `seaborn.pairplot`) 生成类似图 1.4 的代码, 或者用这些特征训练一个分类模型 (如 KNN、决策树、SVM) , 我可以为你演示完整流程。是否继续?

Figure 1.5 (a) First 9 test MNIST gray-scale images. (b) Same as (a), but with the features permuted randomly. Classification performance is identical on both versions of the data (assuming the training data is permuted in an identical way). Figure generated by `shuffledDigitsDemo`.

Now consider the harder problem of classifying images directly, where a human has not preprocessed the data. We might want to classify the image as a whole, e.g., is it an indoors or outdoors scene? is it a horizontal or vertical

photo? does it contain a dog or not? This is called image classification.

In the special case that the images consist of isolated handwritten letters and digits, for example, in a postal or ZIP code on a letter, we can use classification to perform handwriting recognition. A standard dataset used in this area is known as MNIST, which stands for "Modified National Institute of Standards"⁵. (The term "modified" is used because the images have been preprocessed to ensure the digits are mostly in the center of the image.) This dataset contains 60,000 training images and 10,000 test images of the digits 0 to 9, as written by various people. The images are size 28×28 and have grayscale values in the range 0 : 255. See Figure 1.5(a) for some example images.

Many generic classification methods ignore any structure in the input features, such as spatial layout. Consequently, they can also just as easily handle data that looks like Figure 1.5(b), which is the same data except we have randomly permuted the order of all the features. (You will verify this in Exercise 1.1.) This flexibility is both a blessing (since the methods are general purpose) and a curse (since the methods ignore an obviously useful source of information). We will discuss methods for exploiting structure in the input features later in the book.

以下是这段关于图像分类的内容的**中文翻译与详细解释**, 包括图 1.5 的意义:

中文翻译:

图 1.5:

- (a): 前 9 张 MNIST 灰度测试图像;
- (b): 与 (a) 相同, 但**图像像素特征的顺序被随机打乱**;
- 假设训练数据也被**以相同方式打乱**, 则两种数据的分类性能是**相同的**。
- 图像由 `shuffledDigitsDemo` 生成。

现在让我们考虑一个更困难的问题: **直接对图像进行分类**, 在这种情况下, **人类并未预处理数据**。

例如，我们可能想要判断整张图片的内容：

- 是室内场景还是室外？
- 是横向还是纵向照片？
- 是否包含狗？

这类任务被称为 **图像分类** (image classification)。

如果图像中只是**单个手写字符或数字**（比如信件上的邮政编码），我们可以使用分类方法进行**手写识别** (handwriting recognition)。

一个用于这类任务的标准数据集叫做 MNIST，其全称是：

Modified National Institute of Standards

(其中“Modified”表示图像已经被预处理，使得数字基本位于图像中央)

该数据集包括：

- 60,000 张训练图像；
- 10,000 张测试图像；
- 每张图像是一个手写数字 (0~9)，由不同人书写；
- 图像大小为 28×28 像素，灰度值范围为 0 到 255。

你可以在 **图 1.5(a)** 中看到一些示例图像。

许多通用的分类方法（如逻辑回归、KNN、SVM 等）**忽略了输入特征的结构信息**，例如像素的空间布局。

因此，它们在处理如 **图 1.5(b)** 所示的数据时也能正常工作：

- 图像的像素被**随机打乱**（例如，原本是从左上到右下的扫描顺序，现在可能是任意顺序）；
- 只要训练数据也以**相同方式**打乱，模型仍能学习并正确分类。

这种灵活性是：

- **优点**：方法具有通用性（不依赖于输入结构）；
- **缺点**：忽略了输入中明显有用的信息（例如像素的空间关系）。

本书稍后将讨论如何利用输入特征中的结构信息。

解释与重点说明：

概念	说明
图像分类	识别整个图像属于哪个类别（如是否含狗、室内/室外等）
MNIST 数据集	标准的手写数字识别数据集，图像为 28x28 灰度图
预处理内容	数字居中、统一尺寸、灰度标准化
(a) vs. (b)	图像内容一样，但像素顺序在 (b) 中被打乱
分类器的盲点	传统方法不利用像素的空间结构，只看每个像素的数值
启示	更高级的模型（如卷积神经网络 CNN）可以捕捉空间结构，效果更好

衍生思考：为什么打乱像素顺序还能分类？

因为传统分类器只是看“**每个特征值的组合**”是否能预测标签，**并不关心它们代表图像哪个位置的像素**。

但：

- 人眼或 CNN 会利用**位置、邻近关系、形状边缘等空间结构**；
- 这就是深度学习模型优于传统模型的地方。

如果你想：

- 用 Python 可视化 MNIST 图像及其“打乱版”；
- 编写代码对比传统分类器（如逻辑回归）与结构感知方法（如 CNN）；
- 或了解“打乱特征”如何影响训练和泛化，

我可以继续为你演示！是否继续？

Figure 1.6 Example of face detection. (a) Input image (Murphy family, photo taken 5 August 2010). Used with kind permission of Bernard Diedrich of Sherwood Studios.

(b) Output of classifier, which detected 5 faces at different poses. This was produced using the online demo at <http://demo.pittpatt.com/>. The classifier was trained on 1000s of manually labeled images of faces and non-faces, and then was applied to a dense set of overlapping patches in the test image. Only the patches whose probability of containing a face was sufficiently high were returned. Used with kind permission of Pittpatt.com

A harder problem is to find objects within an image; this is called object detection or object localization. An important special case of this is face detection. One approach to this problem is to divide the image into many small overlapping patches at different locations, scales and orientations, and to classify each such patch based on whether it contains face-like texture or not. This is called a sliding window detector. The system then returns those locations where the probability of face is sufficiently high. See Figure 1.6 for an example. Such face detection systems are built-in to most modern digital cameras; the locations of the detected faces are used to determine the center of the auto-focus. Another application is automatically blurring out faces in Google's StreetView system. Having found the faces, one can then proceed to perform face recognition, which means estimating the identity of the person (see Figure 1.10(a)). In this case, the number of class labels might be very large. Also, the features one should use are likely to be different than in the face detection problem: for recognition, subtle differences between faces such as hairstyle may be important for determining identity, but for detection, it is important to be invariant to such details, and to just focus on the differences between faces and non-faces. For more information about visual object detection, see e.g., (Szeliski 2010).

以下是这段关于**人脸检测与对象检测**内容的**中文翻译与解释**, 包括图 1.6 的详细说明:

中文翻译:

图 1.6: 人脸检测的示例

- (a): 输入图像 (Murphy一家, 拍摄于2010年8月5日), 图片经 Bernard Diedrich 授权使用;
- (b): 分类器的输出结果, 检测出了5张处于不同姿态的人脸;
 - 使用了在线演示平台 <http://demo.pittpatt.com>;
 - 该分类器由数千张手动标注的“人脸/非人脸”图像训练而成;
 - 然后应用到测试图像中**密集重叠的小图块 (patches) **上;
 - 最后, 只返回那些被判定为包含人脸的概率足够高的图块。

图像经 Pittpatt.com 授权使用。

一个更难的问题是: 在图像中找出对象的位置。这被称为 **目标检测 (object detection)** 或 **目标定位 (object localization)**。

其中一个重要的特例是: 人脸检测 (face detection)。

解决思路: 滑动窗口检测器 (sliding window detector)

方法如下:

1. 将图像划分为很多重叠的小图块 (patch);
2. 这些图块涵盖不同的位置、尺度和方向;
3. 对每个图块进行分类, 判断其是否具有“类似人脸的纹理”;
4. 返回那些人脸概率足够高的位置作为检测结果。

详见图 1.6。

应用场景举例:

- 数码相机自动对焦: 通过检测人脸, 自动设定对焦中心;
- Google 街景模糊处理: 检测出人脸后进行自动模糊保护隐私;
- 下一步: 人脸识别 (face recognition):
 - 即判断“这是谁”;
 - 所需类别标签可能很多 (数百上千个身份);
 - 识别所需特征不同于检测任务:
 - 识别需要注意细微差别 (如发型、五官、眼镜等);
 - 检测则需要对这些细节保持不变性, 只关注“是否是人脸”。

想了解视觉目标检测的更多信息，请参考：

- Szeliski (2010) 的相关资料。

总结与重点解释：

概念	说明
任务类型	目标检测（如检测人脸在图像中出现的“位置”）
与分类的区别	分类是判断整张图像的类别；检测还需给出“在哪里”
关键技术	滑动窗口检测器：扫描图像中的所有小区域并分类
人脸识别 vs. 人脸检测	识别=“这是谁”；检测=“这是不是脸”
特征设计差异	检测要忽略细节，识别要关注细节
现代应用	数码相机、视频监控、隐私保护（如模糊人脸）
训练数据	大量“人脸/非人脸”图像 + 手工标注位置

延伸说明：

- 滑动窗口方法是传统目标检测的经典思路；
- 它效率不高，但思路简单直接；
- 现代检测算法（如 YOLO、Faster R-CNN）则可同时预测类别和位置，性能更强；
- 人脸识别则是一个多类分类问题（类别数=可能的身份数），常用人脸嵌入向量和相似度比较实现。

如果你想：

- 演示如何用 OpenCV 或 Python 代码实现人脸检测；
- 了解滑动窗口的具体算法；
- 比较传统方法与现代 CNN 检测器的效果，

我可以带你逐步实现或可视化结果。需要吗？

Figure 1.7 (a) Linear regression on some 1d data. (b) Same data with polynomial regression (degree 2). Figure generated by linregPolyVsDegree.

Regression is just like classification except the response variable is continuous. Figure 1.7 shows a simple example: we have a single real-valued input $x_i \in \mathbb{R}$, and a single real-valued response $y_i \in \mathbb{R}$. We consider fitting two models to the data: a straight line and a quadratic function. (We explain how to fit such models below.) Various extensions of this basic problem can arise, such as having high-dimensional inputs, outliers, non-smooth responses, etc. We will discuss ways to handle such problems later in the book.

Here are some examples of real-world regression problems.

Predict tomorrow's stock market price given current market conditions and other possible side information.

Predict the age of a viewer watching a given video on YouTube.

Predict the location in 3d space of a robot arm end effector, given control signals (torques) sent to its various motors.

Predict the amount of prostate specific antigen (PSA) in the body as a function of a number of different clinical measurements.

Predict the temperature at any location inside a building using weather data, time, door sensors, etc.

以下是这段关于**回归 (Regression) **的内容的中文翻译与解释，包括图 1.7 所展示的内容：

中文翻译：

图 1.7：

- (a): 在一维数据上进行**线性回归**；
- (b): 使用同一数据，但应用的是**二次多项式回归**（即二阶）；
- 图像由 `linregPolyVsDegree` 工具生成。

****回归 (Regression) **与分类类似，唯一的区别是响应变量（输出）是连续的。**

图 1.7 展示了一个简单的例子：

- 输入是单一的实数 $x_i \in \mathbb{R}$ ；
- 输出（响应）也是一个实数 $y_i \in \mathbb{R}$ ；
- 我们对这些数据尝试拟合两个模型：
 - 一条直线（线性回归）；
 - 一个二次函数（多项式回归，阶数为 2）；
- （我们将在后文详细解释如何拟合这些模型。）

这个基本问题有很多现实中的扩展形式，例如：

- 高维输入；
 - 异常值 (outliers)；
 - 响应变量不平滑（如噪声大或变化剧烈）；
- 这些情况本书将在后续章节中讨论解决方法。

以下是一些现实中的回归问题示例：

1. 预测明天股市价格，基于当前市场状况和其他辅助信息；
2. 预测 YouTube 视频观众的年龄；
3. 预测机器人手臂末端在三维空间中的位置，给定其各个电机的控制信号（力矩）；
4. 预测体内前列腺特异抗原 (PSA) 水平，基于多种临床指标；
5. 预测建筑物内任意位置的温度，基于天气数据、时间、门禁传感器等信息。

解释与要点总结：

项目	内容
回归定义	学习输入 x 与连续输出 y 之间的映射关系
与分类的区别	分类预测离散标签，回归预测连续值
图 1.7 意义	对比线性回归与多项式回归 (2 阶) 拟合效果差异
多项式回归	更复杂的模型可以拟合非线性趋势，但也可能过拟合

项目	内容
扩展情境	高维输入、异常点、非平滑输出等实际挑战
实际应用	金融预测、生物医学、机器人控制、环境建模等

图像分析（图 1.7）：

- (a) **线性回归**: 拟合一条直线，适合数据大致呈线性趋势时；
- (b) **二次回归**: 拟合一条曲线，更灵活，能捕捉非线性变化；
- 图中展示了**模型复杂度对拟合效果的影响**，为后续讨论**欠拟合与过拟合**打基础。

如果你想进一步了解如何：

- 用 Python 实现线性与多项式回归；
- 可视化模型拟合过程；
- 比较不同阶数对误差与泛化能力的影响；

我可以帮你一步步实现演示。是否继续？

We now consider unsupervised learning, where we are just given output data, without any inputs. The goal is to discover “interesting structure” in the data; this is sometimes called knowledge discovery. Unlike supervised learning, we are not told what the desired output is for each input. Instead, we will formalize our task as one of density estimation, that is, we want to build models of the form $p(x_i|\theta)$. There are two differences from the supervised case. First, we have written $p(x_i|\theta)$ instead of $p(y_i|x_i, \theta)$; that is, supervised learning is conditional density estimation, whereas unsupervised learning is unconditional density estimation. Second, x_i is a vector of features, so we need to create multivariate probability models. By contrast, in supervised learning, y_i is usually just a single variable that we are trying to predict. This means that for most supervised learning problems, we can use univariate probability models (with input-dependent

parameters), which significantly simplifies the problem. (We will discuss multi-output classification in Chapter 19, where we will see that it also involves multivariate probability models.)

Unsupervised learning is arguably more typical of human and animal learning. It is also more widely applicable than supervised learning, since it does not require a human expert to manually label the data. Labeled data is not only expensive to acquire⁶, but it also contains relatively little information, certainly not enough to reliably estimate the parameters of complex models. Geoff Hinton, who is a famous professor of ML at the University of Toronto, has said:

When we're learning to see, nobody's telling us what the right answers are — we just look. Every so often, your mother says "that's a dog", but that's very little information. You'd be lucky if you got a few bits of information — even one bit per second — that way. The brain's visual system has 1014 neural connections. And you only live for 109 seconds. So it's no use learning one bit per second. You need more like 105 bits per second. And there's only one place you can get that much information: from the input itself. — Geoffrey Hinton, 1996 (quoted in (Gorder 2006)).

以下是这段关于**无监督学习 (Unsupervised Learning)** 的中文翻译与解释：

中文翻译：

我们现在来讨论**无监督学习**。在这种学习设置中，我们只有输出数据（如样本特征），**没有对应的输入或标签信息**。

目标是：**在数据中发现“有趣的结构”**，这一过程有时也被称为**“知识发现 (knowledge discovery) ”**。

与监督学习不同，我们不知道每个输入对应的理想输出是什么。因此，我们将任务形式化为**密度估计 (density estimation) **的问题，也就是说，我们希望构建如下形式的模型：

$$p(x_i \mid \theta)$$

与监督学习相比，这里有两个关键区别：

1. 我们写的是 $p(x_i \mid \theta)$ 而不是 $p(y_i \mid x_i, \theta)$:
 - 监督学习是条件密度估计（预测标签），
 - 无监督学习是无条件密度估计（建模样本分布本身）；
2. x_i 是一个特征向量（可能有多维），因此我们需要构建多变量概率模型（multivariate probability models）：
 - 而监督学习中的 y_i 往往是一个单变量，只需单变量概率模型（输入依赖参数），从而使问题更容易。

(我们将在第 19 章讨论多输出分类，它涉及到多变量概率建模。)

无监督学习可以说更符合人类和动物的学习方式，而且也比监督学习更具广泛适用性，因为它**不依赖人工标注数据**。

- 带标签的数据不仅获取成本高，而且信息量有限；
- 对于复杂模型来说，这些标签数据往往不足以支撑可靠的参数估计。

加拿大多伦多大学著名机器学习教授 Geoffrey Hinton 曾说：

“当我们学会看东西时，没人告诉我们正确的答案——我们只是看。
偶尔你妈妈会说‘那是一只狗’，但这提供的信息非常有限。
如果你每秒钟能获得一两比特的信息，那你就算是幸运了。
人脑的视觉系统有 10^{14} 个神经连接，而你一生大约只有 10^9 秒。
所以每秒学一点信息根本不够——你需要每秒学习大约 10^5 比特的信息。
而唯一能提供这么多信息的地方，就是输入本身。”

—— Geoffrey Hinton, 1996 (引自 Gorder, 2006)

✓ 总结与重点解释：

内容	说明
任务目标	在无标签数据中发现模式、结构、分布等信息
建模方式	构建 $p(x_i \mid \theta)$ ，即无条件概率模型
特征维度	x_i 是多维向量，需用多变量概率模型
对比监督学习	无监督学习 ≠ 预测标签，而是建模数据结构本身
优势	无需人工标注、更符合人类的自然学习过程

内容	说明
挑战	模型选择更难，缺乏明确目标函数或误差度量
现实意义	适用于聚类、降维、生成建模、密度估计等任务

📌 衍生任务与方法示例（无监督学习中常见）：

- **聚类 (Clustering)**：如 K-means、层次聚类
- **密度估计**：如 高斯混合模型 (GMM) 、核密度估计
- **降维**：如 主成分分析 (PCA) 、t-SNE、UMAP
- **异常检测**：找出偏离正常分布的样本
- **表示学习**：如自编码器 (Autoencoder) 、对比学习 (Contrastive Learning)

如果你希望我帮你演示一个无监督学习实例（如用 Python 实现聚类或降维），或者更深入理解 Geoffrey Hinton 提到的“从输入中学习”理念，我可以继续为你展开。是否需要？

Figure 1.8 (a) The height and weight of some people. (b) A possible clustering using K = 2 clusters. Figure generated by kmeansHeightWeight.

As a canonical example of unsupervised learning, consider the problem of clustering data into groups. For example, Figure 1.8(a) plots some 2d data, representing the height and weight of a group of 210 people. It seems that there might be various clusters, or subgroups, although it is not clear how many. Let K denote the number of clusters. Our first goal is to estimate the distribution over the number of clusters, $p(K|D)$; this tells us if there are subpopulations within the data. For simplicity, we often approximate the distribution $p(K|D)$ by its mode, $K^* = \arg \max_K p(K|D)$. In the supervised case, we were told that there are two classes (male and female), but in the unsupervised case, we are free to choose as many or few clusters as we like. Picking a model of the “right” complexity is called model selection, and will be discussed in detail below.

Our second goal is to estimate which cluster each point belongs to. Let $z_i \in \{1, \dots, K\}$ represent the cluster to which data point i is assigned. (z_i is an example of a hidden or latent variable, since it is never observed in the training set.) We can infer which cluster each data point belongs to by computing $z^* = \text{argmax}_k p(z_i = k | x_i, D)$. This is illustrated in Figure 1.8(b), where we use different colors to indicate the assignments, assuming $K = 2$.

In this book, we focus on model based clustering, which means we fit a probabilistic model to the data, rather than running some ad hoc algorithm. The advantages of the model-based approach are that one can compare different kinds of models in an objective way (in terms of the likelihood they assign to the data), we can combine them together into larger systems, etc. Here are some real world applications of clustering.

In astronomy, the autoclass system (Cheeseman et al. 1988) discovered a new type of star, based on clustering astrophysical measurements.

In e-commerce, it is common to cluster users into groups, based on their purchasing or web-surfing behavior, and then to send customized targeted advertising to each group (see e.g., (Berkhin 2006)).

In biology, it is common to cluster flow-cytometry data into groups, to discover different sub-populations of cells (see e.g., (Lo et al. 2009)).

以下是这段关于**无监督学习中的聚类 (clustering)** 内容的中文翻译与详细解释，包含图 1.8 的讲解：

中文翻译：

图 1.8：

- (a): 一组人的**身高与体重的二维散点图**；
- (b): 将这些数据聚成 $K = 2$ 个簇 (clusters) 后的示意图；
- 图像由 `kmeansHeightWeight` 工具生成。

作为无监督学习的经典例子，我们考虑将数据**聚类 (clustering) **为若干组的问题。

例如，图 1.8(a) 显示了某组人（共 210 人）的**二维数据**：每个点代表一个人的**身高和体重**。

从图中可以看出数据可能存在一些**簇状结构** (subgroups)，尽管**具体有多少簇不确定**。

我们有两个目标：

🎯 目标一：估计簇的数量 K

- 令 K 表示簇的数量；
- 目标是估计其后验分布：

$$p(K \mid D)$$

- 它告诉我们：数据中是否存在子群体；
- 通常，我们会用该分布的**众数 (mode)** 来近似簇数：

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

- 在监督学习中**，我们可能被明确告知类别数（如男性和女性）；
 - 但在无监督学习中**，我们可以自己选择簇数，或者通过模型选择来确定“合适的复杂度”。
-

🎯 目标二：估计每个数据点属于哪个簇

- 令 $z_i \in \{1, \dots, K\}$ 表示第 i 个样本所属的簇；
- z_i 是一个**隐藏变量 (latent variable)**，因为它在训练数据中是**不可见的**；
- 我们可以估计：

$$z_i^* = \arg \max_k p(z_i = k \mid x_i, D)$$

- 在图 1.8(b) 中展示了这种分配结果，使用不同的颜色标出每个簇，假设 $K = 2$ 。
-

📘 本书采用的聚类方法：基于模型的聚类 (Model-Based Clustering)

这意味着我们不是随意使用某种启发式算法，而是：

- 拟合一个概率模型**（如高斯混合模型 GMM）；

- 可以客观地比较不同模型（依据它们对数据的似然概率）；
 - 模型之间可以组合、嵌套、集成进更大的系统；
 - 适用于复杂、可解释性强的任务。
-

现实中的聚类应用示例：

领域	应用描述
天文学	Autoclass 系统通过对天文测量数据聚类，发现了新类型的恒星 (Cheeseman 等, 1988)
电商推荐	将用户按购买/浏览行为聚类，以便个性化广告推送 (Berkhin, 2006)
生物学	将流式细胞术 (flow-cytometry) 数据聚类，以识别不同的细胞亚群 (Lo 等, 2009)

总结与关键点：

项目	内容
任务类型	无监督聚类 (Unsupervised Clustering)
示例数据	身高与体重组成了二维数据集
图 1.8(a)	原始散点图，未分簇
图 1.8(b)	假设 $K = 2$ 的聚类结果示意
关键变量	K : 簇数； z_i : 每个样本的簇标识
聚类类型	模型驱动型聚类（如 GMM）而非启发式
应用领域	天文、电商、生物、金融、社群分析等

延伸说明：为什么选择“基于模型”的聚类？

- 更具理论基础；
- 可评估模型优劣（如使用 AIC/BIC、对数似然）；
- 易于集成进复杂系统中（如生成模型、变分方法等）；

- 能处理不确定性（如 soft clustering，即每个样本属于某个簇的概率）。

如你希望我带你用 Python 演示：

- 如何用 KMeans 或 GaussianMixture 聚类身高体重数据；
- 如何可视化不同簇数下的结果；
- 或讲解如何评估聚类质量（如轮廓系数、BIC 等），

我可以马上帮你操作！是否继续？

Figure 1.9 (a) A set of points that live on a 2d linear subspace embedded in 3d. The solid red line is the first principal component direction. The dotted black line is the second PC direction. (b) 2D representation of the data.
Figure generated by pcaDemo3d.

Figure 1.10 a) 25 randomly chosen 64×64 pixel images from the Olivetti face database. (b) The mean and the first three principal component basis vectors (eigenfaces).
Figure generated by pcalmageDemo.

When dealing with high dimensional data, it is often useful to reduce the dimensionality by projecting the data to a lower dimensional subspace which captures the “essence” of the data. This is called dimensionality reduction. A simple example is shown in Figure 1.9, where we project some 3d data down to a 2d plane. The 2d approximation is quite good, since most points lie close to this subspace. Reducing to 1d would involve projecting points onto the red line in Figure 1.9(a); this would be a rather poor approximation. (We will make this notion precise in Chapter 12.)

The motivation behind this technique is that although the data may appear high dimensional, there may only be a small number of degrees of variability, corresponding to latent factors. For example, when modeling the appearance of face images, there may only be a few underlying latent factors which describe most of the variability, such as lighting, pose, identity, etc, as illustrated in Figure 1.10.

When used as input to other statistical models, such low dimensional representations often result in better predictive accuracy, because they focus on the “essence” of the object, filtering out inessential features. Also, low dimensional representations are useful for enabling fast nearest neighbor searches and two dimensional projections are very useful for visualizing high dimensional data.

The most common approach to dimensionality reduction is called principal components analysis or PCA. This can be thought of as an unsupervised version of (multi-output) linear regression, where we observe the high-dimensional response y , but not the low-dimensional “cause” z . Thus the model has the form $z \rightarrow y$; we have to “invert the arrow”, and infer the latent low-dimensional z from the observed high-dimensional y . See Section 12.1 for details.

Dimensionality reduction, and PCA in particular, has been applied in many different areas. Some examples include the following:

In biology, it is common to use PCA to interpret gene microarray data, to account for the fact that each measurement is usually the result of many genes which are correlated in their behavior by the fact that they belong to different biological pathways.

In natural language processing, it is common to use a variant of PCA called latent semantic analysis for document retrieval (see Section 27.2.2).

In signal processing (e.g., of acoustic or neural signals), it is common to use ICA (which is a variant of PCA) to separate signals into their different sources (see Section 12.6).

In computer graphics, it is common to project motion capture data to a low dimensional space, and use it to create animations. See Section 15.5 for one way to tackle such problems.

以下是这段关于降维 (Dimensionality Reduction) 的内容的中文翻译与详细解释，涵盖图 1.9 和图 1.10 的说明：

中文翻译：

图 1.9:

- (a): 一组点位于三维空间中的一个**二维线性子空间** ;
 - 红色实线表示**第一主成分方向**;
 - 黑色虚线表示**第二主成分方向**。
 - (b): 这些点被投影到二维平面上的结果。
 - 图由 `pcaDemo3d` 工具生成。
-

图 1.10:

- (a): 来自 Olivetti 人脸数据库中 25 张随机选取的 64×64 像素人脸图像;
 - (b): 平均人脸图像 (mean face) 及前三个主成分基向量 (称为**特征脸 eigenfaces**) ;
 - 图由 `pcaImageDemo` 工具生成。
-

当处理**高维数据**时，通常有必要将其**投影到低维子空间**，以便捕捉数据的“本质”。

这种技术称为**降维 (dimensionality reduction)**。

一个简单例子如图 1.9 所示：

- 原始数据在三维空间;
- 我们将其**投影到二维平面**;
- 这种 2D 近似是合理的，因为大多数数据点靠近该平面;
- 如果降维到 1D，即投影到红线方向（第一主成分），近似效果会很差。

(我们将在第 12 章更严格地定义“近似效果”)

❖ 降维的动机：

虽然数据看起来是高维的，但其变化往往受**少数几个隐藏因子 (latent factors) **控制。

例如在人脸图像建模中，变化可能由以下因素主导：

- 光照 (lighting)
- 姿态 (pose)

- 个人身份 (identity)

如图 1.10 所示。

低维表示的优点：

- **提高预测性能**: 低维特征关注“核心内容”，过滤掉噪声与不必要细节；
 - **更高效率的最近邻搜索**: 用于推荐系统、图像搜索等；
 - **可视化高维数据**: 二维投影便于展示与理解数据结构；
 - **更少参数、更少过拟合风险**。
-

常见方法：主成分分析 (PCA)

- PCA (Principal Component Analysis) 是最常用的降维方法；
- 可视为一种无监督的、多输出线性回归的变体：
 - 我们观察到的是高维响应 y ，但未观测到其低维“成因” z ；
 - 模型为 $z \rightarrow y$ ，我们要**反向推断** z ；

详见第 12.1 节。

降维在多个领域的应用：

领域	应用方式
生物学	PCA 用于分析基因芯片数据，识别多个 共表达通路 中的主导因素
自然语言处理	使用 潜在语义分析 (LSA) —— PCA 的一种变体 —— 用于文档检索 (第 27.2.2 节)
信号处理	使用 独立成分分析 (ICA) (PCA 的变种) 来 信号源分离 (如语音、脑电等, 第 12.6 节)
计算机图形学	将 动作捕捉数据 投影到低维空间后用于动画生成 (见第 15.5 节)

小结与重点概念：

概念	含义
降维目的	提取核心信息，去除冗余，增强可解释性

概念**含义**

图 1.9

降维前后点云的可视化效果 ($3D \rightarrow 2D$)

图 1.10

图像降维示例；特征脸展示了主成分对人脸变化的解释能力

PCA 本质

查找最大方差方向的正交基，进行线性投影

Latent factors

潜在因子控制数据结构与变化，虽不可见，但可被建模

应用优势

降噪、加速、可视化、提高分类/聚类性能

如果你想进一步操作，比如：

- 在 Python 中用 PCA 降维可视化图像或结构化数据；
- 重建图像，展示信息损失；
- 比较 PCA 和 t-SNE、UMAP 等非线性方法；

我可以为你逐步演示。是否继续？

Figure 1.11 A sparse undirected Gaussian graphical model learned using graphical lasso (Section 26.7.2) applied to some flow cytometry data (from (Sachs et al. 2005)), which measures the phosphorylation status of 11 proteins.
Figure generated by ggmLassoDemo.

Sometimes we measure a set of correlated variables, and we would like to discover which ones are most correlated with which others. This can be represented by a graph G , in which nodes represent variables, and edges represent direct dependence between variables (we will make this precise in Chapter 10, when we discuss graphical models). We can then learn this graph structure from data, i.e., we compute $\hat{G} = \text{argmax } p(G|D)$.

As with unsupervised learning in general, there are two main applications for learning sparse graphs: to discover new knowledge, and to get better joint probability density estimators. We now give some examples of each.

Much of the motivation for learning sparse graphical

models comes from the systems biology community. For example, suppose we measure the phosphorylation status of some proteins in a cell (Sachs et al. 2005). Figure 1.11 gives an example of a graph structure that was learned from this data (using methods discussed in Section 26.7.2). As another example, Smith et al. (2006) showed that one can recover the neural “wiring diagram” of a certain kind of bird from time-series EEG data. The recovered structure closely matched the known functional connectivity of this part of the bird brain.

In some cases, we are not interested in interpreting the graph structure, we just want to use it to model correlations and to make predictions. One example of this is in financial portfolio management, where accurate models of the covariance between large numbers of different stocks is important. Carvalho and West (2007) show that by learning a sparse graph, and then using this as the basis of a trading strategy, it is possible to outperform (i.e., make more money than) methods that do not exploit sparse graphs. Another example is predicting traffic jams on the freeway. Horvitz et al. (2005) describe a deployed system called JamBayes for predicting traffic flow in the Seattle area; predictions are made using a graphical model whose structure was learned from data.

以下是关于 **图 1.11 和稀疏图模型学习 (Sparse Graphical Model Learning)** 的中文翻译与解释：

中文翻译：

图 1.11：

- 使用 **Graphical Lasso (图形套索)** 方法 (见第 26.7.2 节) 在某组流式细胞术数据 (Sachs et al. 2005) 上学习得到的**稀疏无向高斯图模型**；
- 数据记录了 11 种蛋白质的**磷酸化状态**；
- 图由 `ggmLassoDemo` 工具生成。

有时我们会测量一组**相互关联的变量**, 希望了解**哪些变量彼此之间有直接依赖关系**。

这种关系可以用一个图 G 表示：

- **节点 (nodes) **代表变量；
- 边 (edges)** 表示变量之间的直接依赖 (具体定义将在第 10 章图模型中给出)；
- 我们的目标是从数据中学习这个图结构，即求：

$$\hat{G} = \arg \max p(G | D)$$

与其他无监督学习任务一样，**学习稀疏图结构**主要有两个目标：

1. **发现新的知识** (结构可解释性)；
2. **提升联合概率密度估计的精度** (建模能力)。

下面分别给出这两个目标的实际应用例子。

🎯 目标 1：发现结构 / 知识发现

- 系统生物学 (Systems Biology)** 是研究稀疏图模型的主要动力来源。
 - 例如：通过测量细胞内某些蛋白的磷酸化状态 (Sachs et al., 2005)，我们可以重建蛋白之间的调控网络，如图 1.11 所示。
 - 又如：Smith 等 (2006) 表明，通过对鸟类脑电图 (EEG) 时序数据建模，可以还原神经连接图，该结构与已知的鸟类大脑功能连接非常吻合。
-

🎯 目标 2：用于建模与预测

在某些情境中，我们不关心图结构的解释意义，只希望它能提供**更好的建模基础**。例如：

1. **金融投资组合管理 (Portfolio Management)**：
 - 任务：建模多个股票之间的协方差；
 - Carvalho 和 West (2007) 表明，学习稀疏图并将其用于交易策略能优于不使用图结构的方法 (盈利更多)；
 2. **交通预测 (Traffic Prediction)**：
 - Horvitz 等 (2005) 提出了名为 JamBayes 的系统，用于预测西雅图的交通流量；
 - 系统基于从历史数据中学习到的图模型结构进行预测。
-

✓ 总结与核心概念：

项目	内容
图模型	节点表示变量，边表示变量间的直接依赖关系
学习目标	给定数据集 D ，学习概率最大的图结构 \hat{G}
Graphical Lasso	学习高斯图模型的一种正则化方法，能够产生稀疏结构
应用领域	生物网络、金融建模、神经科学、交通预测等
稀疏图优点	更具可解释性；避免过拟合；减少计算负担
结构变量 z_i	是隐藏变量（未观测到，但可以推断）
知识发现 vs. 建模预测	可解释性 vs. 准确性/实用性，都是学习图结构的有效目标

💡 延伸说明：

- **图结构可视化**（如图 1.11）有助于理解变量之间的关系；
- **稀疏性的含义**：大多数变量之间没有直接连接，只保留最重要的边；
- **Graphical Lasso** = L1 正则化 + 最大似然估计，用于抑制冗余边、提升可解释性；
- 本质上是在学习一个**高维联合分布**的结构化表达。

如果你希望我演示：

- 如何用 Python（如 `sklearn`, `graphical_models`, `networkx`）实现 Graphical Lasso；
- 如何从数据中学习图结构并可视化；
- 或如何将图结构用于预测或异常检测任务，

我可以带你一步步完成。是否继续？

Figure 1.12 (a) A noisy image with an occluder. (b) An estimate of the underlying pixel intensities, based on a pairwise MRF model. Source: Figure 8 of (Felzenszwalb and Huttenlocher 2006). Used with kind permission of

Pedro Felzenszwalb.

Sometimes we have missing data, that is, variables whose values are unknown. For example, we might have conducted a survey, and some people might not have answered certain questions. Or we might have various sensors, some of which fail. The corresponding design matrix will then have “holes” in it; these missing entries are often represented by NaN, which stands for “not a number”. The goal of imputation is to infer plausible values for the missing entries. This is sometimes called matrix completion. Below we give some example applications.

An interesting example of an imputation-like task is known as image inpainting. The goal is to “fill in” holes (e.g., due to scratches or occlusions) in an image with realistic texture. This is illustrated in Figure 1.12, where we denoise the image, as well as impute the pixels hidden behind the occlusion. This can be tackled by building a joint probability model of the pixels, given a set of clean images, and then inferring the unknown variables (pixels) given the known variables (pixels). This is somewhat like market basket analysis, except the data is real-valued and spatially structured, so the kinds of probability models we use are quite different. See Sections 19.6.2.7 and 13.8.4 for some possible choices.

以下是关于 **图 1.12 和缺失数据填补 (imputation)** 的中文翻译与详细解释：

中文翻译：

图 1.12：

- (a): 一张含有噪声且被遮挡的图像；
- (b): 利用**成对马尔可夫随机场模型** (pairwise MRF model) 估计出的图像原始像素强度；
- 来源: Felzenszwalb 和 Huttenlocher (2006) 第 8 图, 获 Pedro Felzenszwalb 授权使用。

有时候，我们面临的任务中包含**缺失数据** (missing data) ，即某些变量的值是未知的。

示例场景包括：

- 问卷调查中，部分受访者未作答某些问题；
- 多传感器系统中，某些传感器发生故障。

此时，设计矩阵 (design matrix) 中会出现“空洞” (缺失值) ，常用 `NaN` (Not a Number) 表示这些**缺失项**。

我们的目标是对这些缺失值进行**填补** (imputation) ，即推测出**合理的估计值**。这个过程有时也称为：

矩阵补全 (matrix completion)

📷 图像修复 (Image Inpainting) : 一种特殊的填补任务

一个有趣的填补任务是**图像修复**：

- 目标：用**真实、自然的纹理**“填补”图像中的缺口（例如由于划痕、遮挡等造成的）；
- 图 1.12 就展示了这样一个过程：
 - 对图像同时进行**去噪** (denoising) 和 **像素填补** (inpainting) ；
 - 背后的方法是：利用一组**干净图像**建立像素间的**联合概率模型**，再基于已知像素推断出未知像素；
- 这与“购物篮分析” (market basket analysis) 类似：
 - 区别在于：图像数据是**实数值且有空间结构**；
 - 所以使用的概率模型完全不同（如 MRF，马尔可夫随机场）。

更多模型细节见本书第：

- 19.6.2.7 节
 - 13.8.4 节
-

✓ 总结与关键要点：

概念	说明
缺失数据	数据中存在未知项（如传感器失效、问卷空项）
表示方式	用 <code>NaN</code> 标记缺失值

概念**说明**

填补目标

对缺失项推断出**合理、概率一致的值**

图像修复

一种特殊填补任务，目标是**还原被遮挡/损坏的图像区域**

图 1.12(b) 技术

使用**成对马尔可夫随机场 (pairwise MRF) **推断遮挡区域像素

概率模型

在图像修复中，需考虑像素之间的空间依赖关系

与关联分析的区别

图像为实值且有空间结构，购物篮分析为二值、无结构

 **延伸说明：**

- **成对 MRF (Pairwise Markov Random Field)** 是建模图像像素依赖的常用方法；
- 它利用像素间局部关系（例如相邻像素强度差异）来建模整体图像；
- **图像修复方法的关键思想是**：通过建模图像的联合分布 $p(\text{pixels})$ ，在观测像素给定的条件下，估计被遮挡或丢失像素的最可能取值；
- 与此类似的应用还有：
 - 矩阵补全 (Netflix 推荐系统)；
 - 数据清洗；
 - 多模态融合中的信息推断。

如果你希望我演示如何：

- 用 Python 或 OpenCV 进行图像 inpainting；
- 构建一个简单的 MRF 模型推断缺失像素；
- 或实现矩阵补全方法（如 SVD、KNN 填补、深度自编码器），

我可以立即带你操作。是否继续？

Figure 1.13 Example of movie-rating data. Training data is in red, test data is denoted by ?, empty cells are unknown.

Another interesting example of an imputation-like task is known as collaborative filtering. A common example of this concerns predicting which movies people will want to watch based on how they, and other people, have rated movies which they have already seen. The key idea is that the prediction is not based on features of the movie or user (although it could be), but merely on a ratings matrix. More precisely, we have a matrix X where $X(m,u)$ is the rating (say an integer between 1 and 5, where 1 is dislike and 5 is like) by user u of movie m . Note that most of the entries in X will be missing or unknown, since most users will not have rated most movies. Hence we only observe a tiny subset of the X matrix, and we want to predict a different subset. In particular, for any given user u , we might want to predict which of the unrated movies he/she is most likely to want to watch.

In order to encourage research in this area, the DVD rental company Netflix created a competition, launched in 2006, with a \$1M USD prize (see <http://netflixprize.com/>). In particular, they provided a large matrix of ratings, on a scale of 1 to 5, for $\sim 18k$ movies created by $\sim 500k$ users. The full matrix would have $\sim 9 \times 10^9$ entries, but only about 1% of the entries are observed, so the matrix is extremely sparse. A subset of these are used for training, and the rest for testing, as shown in Figure 1.13. The goal of the competition was to predict more accurately than Netflix's existing system. On 21 September 2009, the prize was awarded to a team of researchers known as "BellKor's Pragmatic Chaos". Section 27.6.2 discusses some of their methodology. Further details on the teams and their methods can be found at <http://www.netflixprize.com/community/viewtopic.php?id=1537>.

以下是这段关于 **协同过滤 (Collaborative Filtering)** 和 **Netflix Prize 竞赛** 的中文翻译与详细解释，包括图 1.13 的说明：

中文翻译：

图 1.13：

电影评分数据的示例图。

- 红色区域表示训练数据；

- 问号 ? 表示**需要预测的测试数据**；
 - 空白单元格表示评分未知（缺失值）。
-

另一个有趣的“**填补 (imputation) 类任务**”是：

➡ **协同过滤 (Collaborative Filtering)** 。

一个典型应用是：

预测一个用户可能喜欢哪些电影，基于他们自己以及其他**人对已看电影的评分**。

这里的**关键思想**是：

- 预测**并不依赖于用户或电影的具体特征**（尽管可以使用）；
 - 而仅仅基于一个**评分矩阵 (ratings matrix)**。
-

更准确地说，我们有一个矩阵 X ，其中：

- $X(m, u)$: 表示用户 u 对电影 m 的评分（例如 1 到 5 分，1 表示不喜欢，5 表示喜欢）；
 - 由于大多数用户并没有看过大多数电影，矩阵 X 中的大多数值是缺失的；
 - 因此我们只观测到评分矩阵中的极小一部分；
 - 目标是根据**已知评分**来预测**未知部分**（尤其是某个用户可能喜欢但尚未评分的电影）。
-

🎯 **Netflix Prize 竞赛：**

为了推动这一领域的研究，DVD 租赁公司 Netflix 于 2006 年发起了一项比赛，奖金为 100 万美元。

- 提供了一个非常大的评分矩阵（用户对电影的打分），评分范围 1-5；
- **数据规模：**
 - 约 1.8 万部电影；
 - 约 50 万用户；
 - 矩阵总规模约为 9×10^9 项；
 - 但只有约 1% 的**评分数据**是已知的（极度稀疏）；
- 其中一部分用于**训练**，另一部分用于**测试**（如图 1.13 所示）；

- 比赛目标：比 Netflix 原有系统更准确地进行评分预测。

最终，比赛于 2009 年 9 月 21 日结束，由一个名为“BellKor’s Pragmatic Chaos”的研究团队获胜。

相关方法详见本书 **第 27.6.2 节**，更多讨论请见：

👉 <http://www.netflixprize.com/community/viewtopic.php?id=1537>

✓ 总结与关键概念：

概念	说明
协同过滤	根据用户与物品之间的评分矩阵，预测未知评分
矩阵 X	行是电影，列是用户，元素是评分（1–5）
稀疏性	大部分用户-电影评分是未知的
问题形式	填补缺失值 / 矩阵补全问题 (matrix completion)
图 1.13	红色：训练评分；问号：预测目标；空白：未知部分
Netflix Prize	业界推动大规模推荐系统研究的经典竞赛
获胜团队	BellKor’s Pragmatic Chaos (2009)
方法范式	矩阵分解（如 SVD）、混合模型、集成模型等

❖ 延伸说明：协同过滤方法分类

1. **基于邻居 (Neighborhood-based) :**
 - 用户-用户协同过滤；
 - 项目-项目协同过滤。
2. **基于模型 (Model-based) :**
 - 矩阵分解 (Matrix Factorization) 如 SVD、ALS；
 - 潜在因子模型 (Latent Factor Models) ；
 - 深度学习模型 (如 Autoencoder、Embedding Net) 。

💡 应用现实性：

- 推荐系统核心算法；
 - 广泛应用于：视频平台（YouTube、Netflix）、电商（Amazon）、音乐（Spotify）、社交平台（微博、Facebook）；
 - 实质上是在大规模稀疏矩阵中预测缺失值，与图像修复、缺失值填补等任务密切相关。
-

如果你想我演示：

- 如何用 Python (如 Surprise, LightFM, scikit-learn) 实现协同过滤；
- 如何从稀疏矩阵预测评分；
- 或了解 Netflix Prize 背后的 SVD++, 集成方法、隐语义模型等算法，

我可以为你分步骤实现。是否继续？

Figure 1.14 (a) Illustration of a K-nearest neighbors classifier in 2d for K = 3. The 3 nearest neighbors of test point x_1 have labels 1, 1 and 0, so we predict $p(y = 1|x_1, D, K = 3) = 2/3$. The 3 nearest neighbors of test point x_2 have labels 0, 0, and 0, so we predict $p(y = 1|x_2, D, K = 3) = 0/3$. (b) Illustration of the Voronoi tessellation induced by 1-NN. Based on Figure 4.13 of (Duda et al. 2001).
Figure generated by knnVoronoi.

In commercial data mining, there is much interest in a task called market basket analysis. The data consists of a (typically very large but sparse) binary matrix, where each column represents an item or product, and each row represents a transaction. We set $x_{ij} = 1$ if item j was purchased on the i 'th transaction. Many items are purchased together (e.g., bread and butter), so there will be correlations amongst the bits. Given a new partially observed bit vector, representing a subset of items that the consumer has bought, the goal is to predict which other bits are likely to turn on, representing other items the consumer might be likely to buy. (Unlike collaborative filtering, we often assume there is no missing data in the training data, since we know the past shopping behavior of each customer.)

This task arises in other domains besides modeling purchasing patterns. For example, similar techniques can be used to model dependencies between files in complex software systems. In this case, the task is to predict, given a subset of files that have been changed, which other ones need to be updated to ensure consistency (see e.g., (Hu et al. 2010)).

It is common to solve such tasks using frequent itemset mining, which create association rules (see e.g., (Hastie et al. 2009, sec 14.2) for details). Alternatively, we can adopt a probabilistic approach, and fit a joint density model $p(x_1, \dots, x_D)$ to the bit vectors, see e.g., (Hu et al. 2010). Such models often have better predictive accuracy than association rules, although they may be less interpretable. This is typical of the difference between data mining and machine learning: in data mining, there is more emphasis on interpretable models, whereas in machine learning, there is more emphasis on accurate models.

以下是关于 **图 1.14** 及其涉及的两个主题 —— **K近邻算法 (K-NN) 与 市场篮分析 (Market Basket Analysis)** 的中文翻译与详细解释：

● 中文翻译与解释：

图 1.14 说明：

(a): 二维平面上 ** $K = 3$ 的 K近邻分类器 (K-Nearest Neighbors) **示意图：

- 测试点 x_1 的三个最近邻点的标签为：1, 1, 0
→ 所以预测：

$$p(y = 1 | x_1, D, K = 3) = \frac{2}{3}$$

- 测试点 x_2 的三个最近邻点的标签为：0, 0, 0
→ 所以预测：

$$p(y = 1 | x_2, D, K = 3) = 0$$

(b): 1-NN (最近邻分类器) 所对应的 **Voronoi 图 (Voronoi Tesselation) **示意：

- 平面被划分为区域，每个区域内的点都属于距离它最近的训练样本的类别。
- 图像由 `knnVoronoi` 工具生成，灵感来自 Duda 等人 (2001) 第 4.13 图。

市场篮分析 (Market Basket Analysis) 详解:

市场篮分析 是商业数据挖掘中非常核心的一项任务。

数据格式:

- 数据通常是一个**巨大的稀疏二值矩阵**;
- 每一列对应一个商品，每一行对应一次交易；
- 如果在第 i 次交易中购买了商品 j , 则 $x_{ij} = 1$ 。

分析目标:

- 给定一个顾客当前购买的部分商品（即一个部分观察到的位向量）；
- 预测他/她**可能还会购买哪些商品**；
- 和协同过滤不同，这里通常假设训练数据没有缺失值。

应用举例:

场景	描述
 零售推荐	预测顾客接下来可能购买的商品（如“买了牛奶还会买面包”）
 软件工程	根据修改的文件集合，预测哪些其他文件也需要更新，以保持系统一致性 (Hu et al., 2010)

模型方法对比:

1. 频繁项集挖掘 (Frequent Itemset Mining)

- 基于统计规则：
 - 如：“如果买了 A 和 B, 那通常也会买 C”;
- 产生**关联规则 (Association Rules)**；
- 优点：**直观、可解释性强**；
- 缺点：对稀疏数据鲁棒性较差，精度可能低。

2. 概率建模方法 (Probabilistic Modeling)

- 拟合联合概率分布：

$$p(x_1, \dots, x_D)$$

- 对缺失项进行概率推断；
 - 优点：预测精度更高；
 - 缺点：模型更复杂，可解释性较差；
 - 示例参考：Hu et al. 2010。
-

🔍 对比数据挖掘与机器学习：

方向	侧重点
数据挖掘	偏重可解释性（规则、统计）
机器学习	偏重预测准确性（模型泛化）

✓ 总结：

- K-NN 是一种简单而强大的分类方法，支持概率输出；
 - Voronoi 图 可视化了 1-NN 如何划分特征空间；
 - 市场篮分析 是二进制稀疏数据建模的核心问题，可用于推荐系统、维护依赖管理等；
 - 频繁项集 vs 概率模型 是可解释性与预测性之间的典型权衡。
-

如果你想我：

- 演示 K-NN 或 Voronoi 图的 Python 实现；
- 用 `mlxtend` 或 `apriori` 生成市场篮的关联规则；
- 构建二元变量的概率模型（如朴素贝叶斯或 RBM）；

我可以立即操作示范。是否继续？

In this book, we will be focussing on probabilistic models of the form $p(y|x)$ or $p(x)$, depending on whether we are interested in supervised or unsupervised learning respectively. There are many ways to define such models, but the most important distinction is this: does the model have a fixed number of parameters, or does the number

of parameters grow with the amount of training data? The former is called a parametric model, and the latter is called a nonparametric model. Parametric models have the advantage of often being faster to use, but the disadvantage of making stronger assumptions about the nature of the data distributions. Nonparametric models are more flexible, but often computationally intractable for large datasets. We will give examples of both kinds of models in the sections below. We focus on supervised learning for simplicity, although much of our discussion also applies to unsupervised learning.

这段话主要介绍了**概率模型的分类**，特别是**参数模型 (parametric models) 与非参数模型 (nonparametric models) **之间的关键区别。下面是详细的中文翻译和解释：

● 中文翻译：

在本书中，我们将主要关注如下形式的**概率模型**：

- 若为**监督学习**，模型形式为： $p(y | x)$ (即给定输入 x ，预测输出 y)；
- 若为**非监督学习**，模型形式为： $p(x)$ (即建模输入数据本身的分布)。

构建这种模型有许多方法，但**最关键的区别在于**：

模型的参数数量是固定的，还是随着训练数据的增加而增长？

- 如果模型的参数数量是**固定的**，则称为：**参数模型 (parametric model)**；
- 如果模型的参数数量**随着训练数据的增加而增加**，则称为：**非参数模型 (nonparametric model)**。

各自的优缺点：

类型	优点	缺点
参数模型	通常计算更快；容易实现	对数据分布的假设更强，灵活性较差
非参数模型	更灵活，能捕捉复杂模式	对大数据集计算开销大，可能不切实际

我们将在后文中分别介绍这两类模型的实例。

为了简明起见，本书的讨论**重点聚焦于监督学习**，不过大部分内容也适用于非监督学习场

景。

✓ 概念解释与应用场景：

◆ 参数模型 (Parametric Models)

- 模型假设：数据来自**固定结构**，例如高斯分布、线性函数等；
 - 参数数量不会随着训练样本数量的增加而变化；
 - 示例：
 - 线性回归；
 - 逻辑回归；
 - 朴素贝叶斯；
 - 神经网络（虽然有很多参数，但参数数量是固定的，属于“参数模型”）；
-

◆ 非参数模型 (Nonparametric Models)

- 模型假设更弱，不预设数据的分布形式；
 - 参数数量与**数据规模直接相关**；
 - 更擅长适应数据中的微妙结构和非线性；
 - 示例：
 - K 近邻算法 (K-NN)；
 - 决策树（特别是未剪枝的）；
 - 核密度估计 (KDE)；
 - 高斯过程 (Gaussian Process)；
 - 支持向量机 (SVM) 中的核方法（核函数隐式构造无限维空间）；
-

❖ 比喻理解：

- **参数模型**像是穿一套固定大小的衣服，如果合身，就很好；但若体型变化大，它就不太合适了；
 - **非参数模型**像是穿一套可随身调整的衣服，灵活度高，但穿脱复杂、资源消耗多。
-

⌚ 示例对比：

问题	参数模型	非参数模型
二分类问题	逻辑回归 (Logistic Regression)	K-NN, 支持向量机 (SVM)
密度估计	高斯分布拟合	核密度估计 (KDE)
回归任务	线性/多项式回归	K-NN 回归、高斯过程回归

🔍 总结：

- 本书关注的模型是： $p(y | x)$ (监督学习) 或 $p(x)$ (非监督学习)；
- **参数 vs 非参数** 是机器学习模型设计中的一个核心维度；
- 两者的选择取决于任务需求、数据规模、计算资源和精度要求。

如果你希望我进一步：

- 给出 Python 示例实现参数和非参数模型；
- 进行可视化对比；
- 或说明它们在实际项目（如图像识别、文本分类）中的优劣，

欢迎继续提问，我可以帮助你深入理解或动手实践。

Figure 1.15 (a) Some synthetic 3-class training data in 2d.
 (b) Probability of class 1 for KNN with K = 10. (c)
 Probability of class 2. (d) MAP estimate of class label.
 Figure generated by knnClassifyDemo.

A simple example of a non-parametric classifier is the K nearest neighbor (KNN) classifier. This simply “looks at” the K points in the training set that are nearest to the test input x , counts how many members of each class are in this set, and returns that empirical fraction as the estimate, as illustrated in Figure 1.14. More formally,

$$p(y = c|x, D, K) = \frac{1}{K} \sum_{i \in NK(x, D)} I(y_i = c) \quad (1.2)$$

where $NK(x, D)$ are the (indices of the) K nearest points to x in D and $I(e)$ is the indicator function defined as follows:

$$I(e) = \begin{cases} 1 & \text{if } e \text{ is true} \\ 0 & \text{if } e \text{ is false} \end{cases} \quad (1.3)$$

This method is an example of memory-based learning or instance-based learning. It can be derived from a probabilistic framework as explained in Section 14.7.3. The most common distance metric to use is Euclidean distance (which limits the applicability of the technique to data which is real-valued), although other metrics can be used. Figure 1.15 gives an example of the method in action, where the input is two dimensional, we have three classes, and $K = 10$. (We discuss the effect of K below.) Panel (a) plots the training data. Panel (b) plots $p(y = 1|x, D)$ where x is evaluated on a grid of points. Panel (c) plots $p(y = 2|x, D)$. We do not need to plot $p(y = 3|x, D)$, since probabilities sum to one. Panel (d) plots the MAP estimate $\hat{y}(x) = \operatorname{argmax}_c p(y = c|x, D)$.

A KNN classifier with $K = 1$ induces a Voronoi tessellation of the points (see Figure 1.14(b)). This is a partition of space which associates a region $V(x_i)$ with each point x_i in such a way that all points in $V(x_i)$ are closer to x_i than to any other point. Within each cell, the predicted label is the label of the corresponding training point.

以下是 Figure 1.15 相关段落的中文翻译与详细解释，重点围绕 K 近邻分类器 (KNN)，它作为一种非参数分类方法的应用示例：

中文翻译与解析

◆ 图 1.15 说明：

- (a): 二维空间中的三类合成训练数据；
- (b): 在二维网格上计算的类别 1 的概率 $p(y = 1|x, D)$ ；
- (c): 类别 2 的概率 $p(y = 2|x, D)$ ；
- (d): 根据最大后验概率 (MAP) 规则预测的分类标签 $\hat{y}(x) = \arg \max_c p(y = c|x, D)$ 。

该图由 `knnClassifyDemo` 生成。

◆ K 近邻分类器 (K-Nearest Neighbor, KNN)

KNN 是一个非常典型的**非参数分类器**。它的原理如下：

给定一个测试输入 x , KNN 找到训练集中距离它最近的 **K 个样本点**, 统计这 **K 个样本** 中各类的数量，并以该频率作为概率估计。

公式表示:

$$p(y = c \mid x, D, K) = \frac{1}{K} \sum_{i \in N_K(x, D)} I(y_i = c) \quad (\text{公式 1.2})$$

- $N_K(x, D)$: 表示测试点 x 在训练集 D 中最邻近的 K 个点的下标;
- $I(e)$: 指示函数 (Indicator function) , 定义如下:

$$I(e) = \begin{cases} 1, & \text{如果条件 } e \text{ 为真} \\ 0, & \text{否则} \end{cases} \quad (\text{公式 1.3})$$

分类结果解释:

- 通过计算每个类别的概率，我们可以输出“**概率分布**”而不仅仅是“硬标签”;
- 最后我们通过**最大后验概率估计 (MAP) **进行分类:

$$\hat{y}(x) = \arg \max_c p(y = c \mid x, D)$$

这意味着我们选择概率最高的类别作为最终预测。

记忆式学习方法 (Memory-based / Instance-based Learning)

KNN 属于**记忆型方法**: 它不对数据进行建模，而是在预测时直接查找训练数据中的邻近样本;

- **优点**: 简单直观，无需训练;
 - **缺点**: 对测试时的计算资源依赖重，尤其在大数据集上效率低。
-

距离度量:

KNN 通常使用**欧氏距离** (Euclidean distance) :

$$\text{dist}(x_i, x_j) = \sqrt{\sum_{d=1}^D (x_{id} - x_{jd})^2}$$

这也限制了它主要适用于**实值型输入数据**。

当然，也可以使用**其他距离度量**（如曼哈顿距离、马氏距离等）处理不同类型的特征。

✿ K 值的选择：

- K 越小，模型越“灵活”，但对噪声敏感；
 - K 越大，预测更平滑，但可能过度平均，模糊边界；
 - K = 1 时，KNN 相当于 Voronoi 分区（参考图 1.14(b)），即空间被划分为每个训练点的“势力范围”。
-

📌 总结：

特性	KNN 分类器说明
类型	非参数方法
原理	基于训练集中最近的 K 个样本的多数投票或概率平均
输出	类别概率 $p(y = c x, D)$ 和 MAP 分类结果
优势	实现简单，无需训练，效果稳健
局限	需要大量内存，对距离敏感，预测慢，不易扩展到高维数据（维度灾难）

如你有兴趣，我可以：

- 演示用 Python 实现 KNN 的可视化（与图 1.15 类似）；
- 通过 `sklearn` 创建一个交互式 KNN 分类器；
- 比较不同 K 值下的分类效果。

是否要进一步尝试这些内容？

Figure 1.16 Illustration of the curse of dimensionality. (a) We embed a small cube of side s inside a larger unit cube. (b) We plot the edge length of a cube needed to cover a

given volume of the unit cube as a function of the number of dimensions. Based on Figure 2.6 from (Hastie et al. 2009). Figure generated by curseDimensionality.

The KNN classifier is simple and can work quite well, provided it is given a good distance metric and has enough labeled training data. In fact, it can be shown that the KNN classifier can come within a factor of 2 of the best possible performance if $N \rightarrow \infty$ (Cover and Hart 1967). However, the main problem with KNN classifiers is that they do not work well with high dimensional inputs. The poor performance in high dimensional settings is due to the curse of dimensionality.

To explain the curse, we give some examples from (Hastie et al. 2009, p22). Consider applying a KNN classifier to data where the inputs are uniformly distributed in the D -dimensional unit cube. Suppose we estimate the density of class labels around a test point x by “growing” a hypercube around x until it contains a desired fraction f of the data points. The expected edge length of this cube will be $eD(f) = f^{1/D}$. If $D = 10$, and we want to base our estimate on 10% of the data, we have $e10(0.1) = 0.8$, so we need to extend the cube 80% along each dimension around x . Even if we only use 1% of the data, we find $e10(0.01) = 0.63$: see Figure 1.16. Since the entire range of the data is only 1 along each dimension, we see that the method is no longer very local, despite the name “nearest neighbor”. The trouble with looking at neighbors that are so far away is that they may not be good predictors about the behavior of the input-output function at a given point.

这段内容重点讲解了一个影响机器学习算法性能的重要现象：**维度灾难 (Curse of Dimensionality)**，尤其是它对**K 最近邻 (KNN) 分类器**的负面影响。下面是详细的**中文翻译与解释**：

中文翻译

图 1.16 展示了“维度灾难”的影响：

- (a): 我们在一个大的单位立方体内嵌一个边长为 s 的小立方体；
- (b): 展示了为了覆盖单位立方体中某一固定体积，需要的立方体边长，随着维度数 (D) 的增加而变化的曲线。

该图参考自《The Elements of Statistical Learning》(Hastie 等人, 2009 年),
由 `curseDimensionality` 脚本生成。

KNN 分类器非常简单, 并且在满足以下条件时表现良好:

- 距离度量选择合适;
- 拥有充足的标注训练数据。

事实上, 可以证明如果样本数 $N \rightarrow \infty$, KNN 分类器的错误率可以逼近最佳理论性能的两倍以内 (Cover 和 Hart, 1967)。

但问题在于:

KNN 在 **高维输入数据 (High-dimensional input)** 上的表现非常差。

这主要是因为所谓的“**维度灾难 (curse of dimensionality)**”。

🔍 如何理解维度灾难?

参考 Hastie 等人 (2009, 第 22 页) 中的例子:

假设我们有 D 维单位立方体 (即每个特征在 $[0, 1]$ 范围内), 并且数据点在其中**均匀分布**。

我们打算在一个测试点 x 周围增长一个超立方体 (即每个维度上对称扩展), 直到其包含数据总量的一部分 f 。

为此, 超立方体的边长 $e_D(f)$ 应为:

$$e_D(f) = f^{1/D}$$

📌 举例:

- 若 $D = 10$, 想覆盖 10% 的数据 ($f = 0.1$) :

$$e_{10}(0.1) = 0.8$$

→ 说明在每个维度上要延伸 80%, 才能囊括 10% 的点。

- 若只想用 1% 的数据 ($f = 0.01$) :

$$e_{10}(0.01) = 0.63$$

→ 也就是说，即使只想“看看附近的邻居”，在高维空间中，你也得“看得很远”，这个“邻域”已经不再是局部的了。

! 为什么这很糟糕？

KNN 的基本思想是：附近的点代表了对当前输入行为的好预测。

但在高维空间里：

- 所有点看起来都很“远”；
- “最近的邻居”实际上也可能远离测试点；
- 距离变得没有判别力，噪声反而可能主导预测。

因此：

“局部性”失效了，KNN 的直觉基础被破坏，它预测的可靠性大大降低。

图 1.16 图解说明

- **图 (a)**：展示了在单位立方体中嵌入一个小立方体的直观几何结构；
- **图 (b)**：随着维度 D 增加，为了覆盖固定体积 f ，所需边长迅速逼近单位长度。

例如，若 $f = 0.1$ ，边长 $e_D(f)$ 快速趋近 1，这意味着“局部”需要覆盖整个空间，根本称不上是“邻近”。

总结：维度灾难的影响

项目	说明
问题	在高维空间中，“邻近”变得模糊，局部性失效
影响	KNN 等依赖距离度量的模型性能下降严重
数学直觉	维度越高，要覆盖相同的数据比例，局部体积的边长必须变长，接近 1
后果	近邻点可能不再代表测试点的特征，预测误差增大
应对方法	可使用降维、特征选择、使用模型对高维不敏感（如树模型、深度网络）

如果你希望，我可以进一步：

- 画出图 1.16 中的 $e_D(f)$ 曲线；
- 展示 KNN 在不同维度下的实际分类误差；
- 解释如何用 PCA 等方法缓解维度灾难。

是否需要继续深入？

The main way to combat the curse of dimensionality is to make some assumptions about the nature of the data distribution (either $p(y|x)$ for a supervised problem or $p(x)$ for an unsupervised problem). These assumptions, known as inductive bias, are often embodied in the form of a parametric model, which is a statistical model with a fixed number of parameters. Below we briefly describe two widely used examples; we will revisit these and other models in much greater depth later in the book.

这段话介绍了应对“维度灾难（curse of dimensionality）”的主要策略，并引出 **参数模型**（parametric models）的重要性。下面是逐句的**中文翻译与解释**：

中文翻译：

应对维度灾难的主要方法是：

对数据的分布做出某些假设（对于监督学习是 $p(y|x)$ ，对于非监督学习是 $p(x)$ ）。

这些假设被称为**归纳偏置**（inductive bias），它们通常体现为：

一个参数模型（parametric model），即参数数量是固定的统计模型。

下面我们将简要介绍两个被广泛使用的例子；本书后面会深入讨论这些模型和其他更多模型。

解释要点：

术语	含义
维度灾难	高维空间中数据变稀疏、距离失效，导致学习困难

术语	含义
归纳偏置 (Inductive Bias)	算法对假设空间的限制，用于更有效地学习
参数模型 (Parametric Model)	模型结构固定，参数数量不随训练数据量增加而增长。例如线性回归、朴素贝叶斯、神经网络（固定架构）等
非参数模型	参数数量随数据增长而增长，如 KNN、高斯过程等，适合灵活建模但计算成本高

✓ 小结

为应对维度灾难，我们不可能在“完全没有假设”的条件下高效地学习；因此，我们需要引入 **结构性假设 (inductive bias)**，这在机器学习中通常通过选择合适的模型结构（如参数模型）来实现。

下一步通常会介绍两个典型的参数模型，例如：

1. **朴素贝叶斯 (Naive Bayes)**
2. **线性/逻辑回归 (Linear / Logistic Regression)**

这些模型都具有：

- 固定的参数数量；
- 对数据分布的简化假设；
- 易于训练和推理；
- 不易受到维度灾难的影响（相对于非参数方法）。

如果你希望，我可以：

- 具体展开这两个模型的原理和示例；
- 对比参数模型与非参数模型的优缺点；
- 给出实际应用中的模型选择建议。

需要继续吗？

Figure 1.17 (a) A Gaussian pdf with mean 0 and variance 1.
Figure generated by gaussPlotDemo. (b) Visualization of

the conditional density model $p(y|x, \theta) = N(y|w_0 + w_1x, \sigma^2)$. The density falls off exponentially fast as we move away from the regression line. Figure generated by linregWedgeDemo2.

Figure 1.18 Polynomial of degrees 14 and 20 fit by least squares to 21 data points. Figure generated by linregPolyVsDegree.

One of the most widely used models for regression is known as linear regression. This asserts that the response is a linear function of the inputs. This can be written as follows:

$$y(x) = w^T x + \epsilon \quad (1.4)$$

where $w^T x$ represents the inner or scalar product between the input vector x and the model's weight vector w , and ϵ is the residual error between our linear predictions and the true response.

We often assume that ϵ has a Gaussian or normal distribution. We denote this by $\sim N(\mu, \sigma^2)$, where μ is the mean and σ^2 is the variance (see Chapter 2 for details). When we plot this distribution, we get the well-known bell curve shown in Figure 1.17(a).

To make the connection between linear regression and Gaussians more explicit, we can rewrite the model in the following form:

$$p(y|x, \theta) = N(y|\mu(x), \sigma^2(x)) \quad (1.5)$$

This makes it clear that the model is a conditional probability density. In the simplest case, we assume μ is a linear function of x , so $\mu = w^T x$, and that the noise is fixed, $\sigma^2(x) = \sigma^2$. In this case, $\theta = (w, \sigma^2)$ are the parameters of the model.

For example, suppose the input is 1 dimensional. We can represent the expected response as follows:

$$\mu(x) = w_0 + w_1 x = w^T x \quad (1.6)$$

where w_0 is the intercept or bias term, w_1 is the slope, and where we have defined the vector $x = (1, x)$.

(Appending a constant 1 term to an input vector is a common notational trick which allows us to combine the intercept term with the other terms in the model.) If w_1 is positive, it means we expect the output to increase as the input increases. This is illustrated in 1d in Figure 1.17(b); a more conventional plot, of the mean response vs x , is shown in Figure 1.7(a).

Linear regression can be made to model non-linear relationships by replacing x with some non-linear function of the inputs, $\varphi(x)$. That is, we use

$$p(y|x, \theta) = N(y|w^T\varphi(x), \sigma^2) \quad (1.7)$$

This is known as basis function expansion. For example, Figure 1.18 illustrates the case where $\varphi(x) = [1, x_1, x_2, \dots, x_d]$, for $d = 14$ and $d = 20$; this is known as polynomial regression. We will consider other kinds of basis functions later in the book. In fact, many popular machine learning methods — such as support vector machines, neural networks, classification and regression trees, etc. — can be seen as just different ways of estimating basis functions from data, as we discuss in Chapters 14 and 16.

这段内容介绍了**线性回归** (linear regression) 的**基本原理**, 并逐步扩展到**高斯分布建模**、**非线性映射** (如多项式回归) 、以及**模型泛化能力**的问题。下面是详细的中文翻译和解释。

中文翻译与解释:

1 线性回归基本形式

“最广泛使用的回归模型之一是**线性回归**。”

线性回归假设目标变量 y 是输入 x 的线性函数, 即:

$$y(x) = w^T x + \varepsilon = \sum_{j=1}^D w_j x_j + \varepsilon \quad (1.4)$$

- w : 权重向量 (模型参数)
- x : 输入特征向量
- ε : 误差项 (或噪声)

其中误差项通常假设符合**高斯分布** $\varepsilon \sim N(0, \sigma^2)$, 也就是所谓的“正态分布”。

2 用条件概率密度形式表达回归模型

回归模型也可以写成如下形式:

$$p(y|x, \theta) = N(y|\mu(x), \sigma^2) \quad (1.5)$$

- $\mu(x) = w^T x$: 均值 (预测值)
 - σ^2 : 固定方差
 - $\theta = (w, \sigma^2)$: 模型参数集合
- 👉 这表示在输入 x 下，输出 y 服从某个高斯分布。
-

3 线性模型的图像解释

图 1.17(a): 展示了一个标准高斯分布的“钟形曲线”。

图 1.17(b): 展示了二维中 $y = w_0 + w_1 x$ 下的高斯密度 (密度沿回归线快速下降)。

4 非线性建模：基函数展开 (Basis Function Expansion)

线性模型虽然结构简单，但可以通过“变换输入”来建模非线性关系：

$$p(y|x, \theta) = N(y|w^T \phi(x), \sigma^2) \quad (1.7)$$

- $\phi(x)$: 对原始输入 x 的非线性变换 (称为“基函数”)

👉 举例：多项式回归

$$\phi(x) = [1, x, x^2, \dots, x^d]$$

图 1.18 展示了当 $d = 14$ 和 $d = 20$ 时的拟合效果。

👉 这种变换虽然仍使用线性模型 (对参数是线性的)，但整体函数对输入 x 是非线性的。

5 更广义的视角：ML 中很多方法都可视为非线性映射

包括：

- 支持向量机 (SVM)
- 神经网络
- 决策树 (CART)

这些都可看作在学习输入空间的非线性变换 $\phi(x)$ ，然后在变换后的空间中进行线性预测。

✓ 总结

概念	内容
线性回归	假设 $y = w^T x + \varepsilon$, 误差服从高斯分布
概率视角	模型变成条件概率密度 ($p(y x)$)
非线性扩展	使用基函数 $\phi(x)$ 将输入非线性变换后, 再线性拟合
多项式回归	一种常见的基函数形式, 容易导致过拟合 (尤其当度数过高时)

如果你需要我可进一步讲解以下内容:

- 多项式回归中的过拟合问题 (见图 1.18)
- 高斯分布建模在回归中的优点
- 如何选择合适的基函数或模型复杂度 (偏差-方差权衡)

是否要继续?

Figure 1.19 (a) The sigmoid or logistic function. We have $\text{sigm}(-\infty)=0$, $\text{sigm}(0) = 0.5$, and $\text{sigm}(\infty)=1$. Figure generated by sigmoidPlot. (b) Logistic regression for SAT scores. Solid black dots are the data. The open red circles are the predicted probabilities. The green crosses denote two students with the same SAT score of 525 (and hence same input representation x) but with different training labels (one student passed, $y = 1$, the other failed, $y = 0$). Hence this data is not perfectly separable using just the SAT feature. Figure generated by logregSATdemo.

We can generalize linear regression to the (binary) classification setting by making two changes. First we replace the Gaussian distribution for y with a Bernoulli distribution⁹, which is more appropriate for the case when the response is binary, $y \in \{0, 1\}$. That is, we use $p(y|x, w) = \text{Ber}(y|\mu(x))$ (1.8) where $\mu(x) = E[y|x] = p(y=1|x)$. Second, we compute a linear combination of the inputs, as before, but then we pass this through a function that ensures $0 \leq \mu(x) \leq 1$ by defining

$$\mu(x) = \text{sigm}(w^T x) \quad (1.9)$$

where $\text{sigm}(\eta)$ refers to the sigmoid function, also known as the logistic or logit function. This is defined as

$$\text{sigm}(\eta) = \frac{1}{1 + \exp(-\eta)} = e^\eta / (e^\eta + 1) \quad (1.10)$$

The term "sigmoid" means S-shaped: see Figure 1.19(a) for a plot. It is also known as a squashing function, since it maps the whole real line to $[0, 1]$, which is necessary for the output to be interpreted as a probability.

Putting these two steps together we get

$$p(y|x, w) = \text{Ber}(y|\text{sigm}(w^T x)) \quad (1.11)$$

This is called logistic regression due to its similarity to linear regression (although it is a form of classification, not regression!).

A simple example of logistic regression is shown in Figure 1.19(b), where we plot

$$p(y_i = 1|x_i, w) = \text{sigm}(w_0 + w_1 x_i) \quad (1.12)$$

where x_i is the SAT10 score of student i and y_i is whether they passed or failed a class. The solid black dots show the training data, and the red circles plot $p(y_i = 1|x_i, \hat{w})$, where \hat{w} are the parameters estimated from the training data (we discuss how to compute these estimates in Section 8.3.4).

If we threshold the output probability at 0.5, we can induce a decision rule of the form

$$\hat{y}(x) = 1 \iff p(y = 1|x) > 0.5 \quad (1.13)$$

By looking at Figure 1.19(b), we see that $\text{sigm}(w_0 + w_1 x) = 0.5$ for $x \approx 545 = x^*$. We can imagine drawing a vertical line at $x = x^*$; this is known as a decision boundary. Everything to the left of this line is classified as a 0, and everything to the right of the line is classified as a 1.

We notice that this decision rule has a non-zero error rate even on the training set. This is because the data is not linearly separable, i.e., there is no straight line we can draw to separate the 0s from the 1s. We can create models with non-linear decision boundaries using basis function expansion, just as we did with non-linear regression. We will see many examples of this later in the book.

这段内容介绍了逻辑回归 (logistic regression)，它是将线性回归推广到二分类任务中的方法。逻辑回归属于分类算法，而不是“回归”，尽管名字里带“回归”。以下是详细的中文翻译与解释：

中文翻译与分段解释

1 逻辑回归的目标

我们希望建模：

$$p(y|x, w) = \text{Ber}(y|\mu(x))$$

其中：

- $y \in \{0, 1\}$: 标签是二元的 (例如通过/未通过)
- $\mu(x) = p(y = 1|x)$: 是预测为类别1的概率

2 将线性组合输入转为概率：Sigmoid 函数

我们不能直接用线性组合 $w^T x$ 作为概率，因为它可以超出 [0,1] 范围。

所以引入 sigmoid 函数 (又叫 logistic 函数) :

$$\mu(x) = \text{sigm}(w^T x) = \frac{1}{1 + e^{-w^T x}} \quad (1.9)$$

★ Sigmoid 的性质：

- 输入趋于负无穷：输出趋于 0
- 输入为 0：输出是 0.5
- 输入趋于正无穷：输出趋于 1

图 1.19(a) 就是 sigmoid 曲线 (典型的“S形”函数)

3 完整的逻辑回归模型公式

$$p(y|x, w) = \text{Ber}(y|\text{sigm}(w^T x)) \quad (1.11)$$

这意味着预测的是 $y = 1$ 的概率，基于特征 x 和参数 w 。

4 逻辑回归应用实例：SAT 成绩预测

图 1.19(b) 是逻辑回归在 SAT 成绩预测上的例子：

- 黑点：真实训练样本 (SAT分数 + 是否通过)
- 红圈：逻辑回归模型预测的概率 (sigmoid 输出)
- 绿叉：两个学生 SAT 分数一样 (525)，但结果不同 (一个通过，一个未通过)

说明：数据 **不可线性分割**（线性模型无法完全分类训练数据）

5 基于概率进行分类预测

使用以下规则来预测类别：

$$\hat{y}(x) = \begin{cases} 1, & \text{如果 } p(y=1|x) > 0.5 \\ 0, & \text{否则} \end{cases} \quad (1.13)$$

这个0.5阈值在 sigmoid 函数下刚好对应于 $w^T x = 0$ 。

6 决策边界

在图 1.19(b) 中：

- Sigmoid 输出为 0.5 时的 $x \approx 545$, 即是分类的**决策边界**
 - 所有小于 545 的预测为 0 (未通过)
 - 所有大于 545 的预测为 1 (通过)
-

7 非线性可分怎么办？

由于数据并不是完全线性可分，我们可以用和非线性回归类似的方法——**基函数扩展**
(basis expansion) 来引入非线性：

- 例如将 x 转换为 $[1, x, x^2, \dots]$
 - 或使用神经网络、核方法等更强表达能力的方法（后面章节会详细介绍）
-

✓ 总结

概念	内容
逻辑回归	对 ($p(y=1$
输出分布	使用 Bernoulli 分布而非 Gaussian 分布
sigmoid 函数	将线性输出压缩到 $[0,1]$ 区间，便于表示概率
决策边界	sigmoid 输出为 0.5 时的输入点，对应于 $w^T x = 0$
分类规则	$\hat{y}(x) = 1$ 当 ($p(y=1$

如果你希望，我可以继续讲解逻辑回归的 **参数估计**（比如最大似然估计、梯度下降）或如何扩展到多类分类问题（softmax 回归）。是否继续？

When we fit highly flexible models, we need to be careful that we do not overfit the data, that is, we should avoid trying to model every minor variation in the input, since this is more likely to be noise than true signal. This is illustrated in Figure 1.18(b), where we see that using a high degree polynomial results in a curve that is very “wiggly”. It is unlikely that the true function has such extreme oscillations. Thus using such a model might result in accurate predictions of future outputs.

As another example, consider the KNN classifier. The value of K can have a large effect on the behavior of this model. When $K = 1$, the method makes no errors on the training set (since we just return the labels of the original training points), but the resulting prediction surface is very “wiggly” (see Figure 1.20(a)). Therefore the method may not work well at predicting future data. In Figure 1.20(b), we see that using $K = 5$ results in a smoother prediction surface, because we are averaging over a larger neighborhood. As K increases, the predictions becomes smoother until, in the limit of $K = N$, we end up predicting the majority label of the whole data set. Below we discuss how to pick the “right” value of K .

这段内容讲的是机器学习中一个非常重要的概念：**过拟合 (overfitting)**，并以多项式回归和 K 近邻分类器 (KNN) 为例来说明。

💬 中文解释与翻译

✓ 什么是过拟合？

当我们使用过于灵活的模型时，比如：

- 高阶多项式回归（比如14阶或20阶多项式）
- K值很小的KNN（比如K=1）

模型可能会尝试拟合训练数据中的所有细节和噪声，这会导致它在未见过的新数据上表现很差，即泛化能力差。

图 1.18(b): 高阶多项式回归 = 曲线太“抖”

该图展示了使用高阶多项式拟合训练数据时的情况：

- 模型会“穿过”所有训练点，但在中间位置波动剧烈
 - 这种波动往往反映的是数据中的噪声而不是有意义的模式
 - 因此，这样的模型尽管在训练集上表现完美，但在测试集上可能预测错误
-

KNN 也会过拟合

对于 KNN：

- K=1：
 - 每个测试点都取最近的一个训练点的标签
 - 在训练集上不会有错误（训练精度=100%）
 - 但预测表面会**非常“抖动”**
 - 这也表示模型对噪声非常敏感，泛化能力差
 - K=5（如图 1.20(b) 所示）：
 - 每个测试点取最近5个点的多数标签
 - 得到的预测表面较**平滑**
 - K=N（训练样本总数）：
 - 所有预测都取训练集中**最多的标签**，即预测结果完全一样
 - 模型**太简单，欠拟合**
-

如何选择合适的模型复杂度（比如 K 值）？

这就涉及到一个核心概念：

我们需要在“过拟合”与“欠拟合”之间做权衡。

这通常通过：

- **交叉验证 (cross-validation) **来选出最优的 K 或模型复杂度
 - **正则化 (regularization) **来抑制模型过于灵活
 - **偏差-方差权衡 (bias-variance tradeoff) **来理解模型行为 (后续章节会深入讲)
-

总结

概念	描述
过拟合	模型太复杂，记住了训练数据中的噪声，泛化差
欠拟合	模型太简单，无法捕捉到数据中的真实规律
高阶多项式	能拟合复杂曲线，但容易过拟合
小 K 的 KNN	预测准确但对噪声敏感，结果“抖动”
大 K 的 KNN	更平滑，但可能忽略局部结构
解决方法	交叉验证、正则化、合理选模型

需要我用图表演示一下 KNN 不同 K 值下的预测边界变化，或解释如何用交叉验证选择最佳模型吗？

Figure 1.21 (a) Misclassification rate vs K in a K-nearest neighbor classifier. On the left, where K is small, the model is complex and hence we overfit. On the right, where K is large, the model is simple and we underfit. Dotted blue line: training set (size 200). Solid red line: test set (size 500). (b) Schematic of 5-fold cross validation. Figure generated by knnClassifyDemo.

When we have a variety of models of different complexity (e.g., linear or logistic regression models with different degree polynomials, or KNN classifiers with different values of K), how should we pick the right one? A natural approach is to compute the misclassification rate on the training set for each method. This is defined as follows:

$$\text{err}(f, D) = \frac{1}{N} \sum_{i=1}^N I(f(x_i) \neq y_i) \quad (1.14)$$

where $f(x)$ is our classifier. In Figure 1.21(a), we plot this error rate vs K for a KNN classifier (dotted blue line). We see that increasing K increases our error rate on the training set, because we are over-smoothing. As we said above, we can get minimal error on the training set by using $K = 1$, since this model is just memorizing the data. However, what we care about is generalization error, which is the expected value of the misclassification rate when averaged over future data (see Section 6.3 for details). This can be approximated by computing the misclassification rate on a large independent test set, not used during model training. We plot the test error vs K in Figure 1.21(a) in solid red (upper curve). Now we see a U-shaped curve: for complex models (small K), the method overfits, and for simple models (big K), the method underfits. Therefore, an obvious way to pick K is to pick the value with the minimum error on the test set (in this example, any value between 10 and 100 should be fine). Unfortunately, when training the model, we don't have access to the test set (by assumption), so we cannot use the test set to pick the model of the right complexity.¹¹ However, we can create a test set by partitioning the training set into two: the part used for training the model, and a second part, called the validation set, used for selecting the model complexity. We then fit all the models on the training set, and evaluate their performance on the validation set, and pick the best. Once we have picked the best, we can refit it to all the available data. If we have a separate test set, we can evaluate performance on this, in order to estimate the accuracy of our method. (We discuss this in more detail in Section 6.5.3.)

Often we use about 80% of the data for the training set, and 20% for the validation set. But if the number of training cases is small, this technique runs into problems, because the model won't have enough data to train on, and we won't have enough data to make a reliable estimate of the future performance.

A simple but popular solution to this is to use cross validation (CV). The idea is simple: we split the training data into K folds; then, for each fold $k \in \{1, \dots, K\}$, we train on all the folds but the k 'th, and test on the k 'th, in a round-robin fashion, as sketched in Figure 1.21(b). We

then compute the error averaged over all the folds, and use this as a proxy for the test error. (Note that each point gets predicted only once, although it will be used for training $K-1$ times.) It is common to use $K=5$; this is called 5-fold CV. If we set $K=N$, then we get a method called leave-one out cross validation, or LOOCV, since in fold i , we train on all the data cases except for i , and then test on i . Exercise 1.3 asks you to compute the 5-fold CV estimate of the test error vs K , and to compare it to the empirical test error in Figure 1.21(a).

Choosing K for a KNN classifier is a special case of a more general problem known as model selection, where we have to choose between models with different degrees of flexibility. Crossvalidation is widely used for solving such problems, although we will discuss other approaches later in the book.

这段内容深入讲解了 **如何选择模型复杂度**，特别是以 KNN 中选择合适的 K 为例，同时引入了一个核心技术：**交叉验证 (cross-validation)**。

核心概念解释 (中文)

1. 训练误差 vs 测试误差

- 图 1.21(a) 展示了误分类率 (misclassification rate) 随 K 值的变化情况：
 - 蓝色虚线 = **训练集误差**: 当 K 较小时误差最小 (如 $K=1$)，因为模型记住了所有训练数据 (过拟合)
 - 红色实线 = **测试集误差**: 表现为 U 形曲线:
 - 左边 (K 小): 过拟合, 泛化差
 - 右边 (K 大): 欠拟合, 模型太“懒”
 - 中间 (例如 $K \approx 10$ 到 100): 平衡点, 误差最小

2. 问题: 测试集不能用于模型选择!

- 虽然我们想根据测试误差选出最优模型，但**不能用测试集来做选择**，因为：
 - 模型是基于训练数据学出来的
 - 一旦你用测试集参与选择过程，它就不再是一个“干净”的评估标准了

3. 解决方法：验证集 & 交叉验证 (Cross Validation)

验证集 (Validation Set)

- 把训练数据再划分成两部分：
 - **训练子集**: 用于训练模型 (比如用来训练不同的 K 值)
 - **验证集**: 用于比较不同模型在未见数据上的表现，从而**选择最优模型**
- 通常的分配比例是：80% 用于训练，20% 用于验证

问题：当样本较小时，验证集会让训练数据减少，模型也不准

交叉验证 (Cross-Validation)

什么是 K 折交叉验证 (K-fold CV) ?

- 将训练数据分成 K 个“折” (fold)
- 每次：
 - 选择其中 1 个作为验证集，其余 K-1 个作为训练集
 - 对每个 K 都重复训练与验证
- 最终得到每个模型 (比如每个 K 值) 在所有折上的**平均误差**
- 用这个平均值来选最优模型

最常用的配置：

- K=5: 5-fold CV (如图 1.21(b))
 - K=N: 留一法 (LOOCV, Leave-One-Out Cross Validation)
-

总结表格

方法	说明	优缺点
训练误差最小	看起来表现好，但可能严重过拟合	<input checked="" type="checkbox"/> 不能代表泛化能力
测试误差最小	是最终目标，但不能用来选模型	<input checked="" type="checkbox"/> 会泄露测试信息
验证集	简单实用，常用80/20划分	适合样本多的情况
交叉验证	更稳定、全面评估模型表现	<input checked="" type="checkbox"/> 特别适合小样本情况

✖ 举例：选 K 值的流程 (KNN)

1. 划分数据 → 如果数据量小就用交叉验证
 2. 针对不同的 K 值 (如 K = 1 到 100) 重复：
 - 用交叉验证评估误差
 3. 选择验证误差最小的 K (如图中 K=10 到 100)
 4. 用**所有训练数据**再训练一个最终模型
 5. 最后，用测试集评估模型效果 (但不参与选择过程)
-

如果你想，我可以：

- 用 Python 演示一次 KNN + 交叉验证的完整流程
- 或者画出图 1.21(a)(b) 的模拟图

要来试一下吗？

Much of machine learning is concerned with devising different models, and different algorithms to fit them. We can use methods such as cross validation to empirically choose the best method for our particular problem. However, there is no universally best model — this is sometimes called the no free lunch theorem (Wolpert 1996). The reason for this is that a set of assumptions that works well in one domain may work poorly in another. As a consequence of the no free lunch theorem, we need to develop many different types of models, to cover the wide variety of data that occurs in the real world. And for each model, there may be many different algorithms we can use to train the model, which make different speed-accuracy-complexity tradeoffs. It is this combination of data, models and algorithms that we will be studying in the subsequent chapters.

这段内容总结了机器学习的一个重要哲学观点：**没有万能的模型 (No Free Lunch Theorem)**。以下是对其核心思想的解释：

核心思想解读 (中文)

没有“通用最优模型”

- 在现实中，不存在一个模型可以在所有任务或所有数据集上都表现最好。
- 这被称为“无免费午餐定理”(No Free Lunch Theorem)，由 Wolpert 于 1996 年提出。

为什么？

- 不同的问题、不同的数据结构，对模型的需求也不同。
 - 有些问题需要解释性强的模型（比如医学）
 - 有些问题需要预测准确度更高的模型（比如广告点击率预测）
 - 一个在文本分类中效果很好的模型，可能在图像识别上根本不适用
-

那我们该怎么办？

策略 1：构建多种模型

- 针对不同的数据类型（文本、图像、时间序列等），研究不同的模型结构
- 举例：
 - 文本：朴素贝叶斯、Transformer
 - 图像：CNN、ViT
 - 表格：树模型 (XGBoost)、TabNet

策略 2：使用交叉验证来选模型

- 通过如 K 折交叉验证 (K-fold CV) 来比较不同模型在当前任务上的表现
- 避免“主观偏好”或“模型崇拜”，用数据来决定谁适合当前问题

策略 3：理解训练算法的权衡

- 每种模型训练方式可能有不同的：
 - 速度**：训练时间、推理速度
 - 精度**：泛化误差
 - 复杂度**：实现和调参的难度

总结一句话

没有一种模型是放之四海皆准的。

所以，**理解数据本身、尝试多种模型、比较其表现**，是机器学习实践的核心。

如果你有兴趣，我可以：

- 详细举几个“不同模型在不同任务上优劣对比”的例子
- 用图表帮你总结模型/算法选择的通用框架
- 或者结合你自己的项目，给出合适的模型建议

你想怎么继续？

K折交叉验证里的不同模型是指什么，是相同算法使用不同的训练集训练出的不同模型吗

这是一个非常关键的问题！我们来仔细区分一下你说的两种情况，并说明在 K 折交叉验证 (K-fold cross-validation) 中，“不同模型”具体指什么。

简明回答：

在 K 折交叉验证中，“不同模型”指的是相同算法在不同训练子集上训练出来的不同实例，即 **相同算法 + 不同训练集 → 不同模型实例**。

这些模型都是用来估计这个算法的泛化能力（比如误差率）而产生的。它们**并不是用来比较算法优劣的“候选模型”**，而是为了评估一个给定算法的表现。

举例说明

假设我们想用 **KNN 算法**、 $K=5$ 来解决一个分类问题，并用 **5 折交叉验证** 来评估它的性能。

步骤：

1. 将训练数据集划分成 5 个不重叠的“folds”
2. 重复 5 次：
 - 每次选择 1 个 fold 作为验证集，其余 4 个作为训练集

- 用这 4 个 folds 训练一个 KNN 模型实例
 - 用该模型在验证集上做预测并计算错误率
3. 将 5 次的验证误差进行平均，得到这个 KNN 算法在该数据集上的泛化误差估计

所以这里训练出了 5 个模型实例，但它们是：

- 相同的算法结构 (KNN, K=5)
- 不同的训练子集
- 用来估算算法整体表现，不用于预测新数据

？和“模型选择”有什么关系？

交叉验证可以作为 模型选择的一部分，例如：

你想比较以下几个模型谁更好：

- KNN (K=3)
- KNN (K=5)
- 决策树 (最大深度 = 4)
- 逻辑回归

那么对每个候选模型结构，你都进行 K 折交叉验证，得到它的平均验证误差，最终选择误差最小的那个模型结构。

这时候你做了：

多个模型结构 × 多次交叉验证 → 比较模型结构优劣

✓ 总结一句话

K 折交叉验证中的“不同模型”是指 相同算法结构 在 不同训练集划分下 训练得到的 多个 模型实例，它们是为了估计该算法的泛化性能而存在的。

如果你是在做“模型选择”，则需要对 多个候选算法/结构 各自做交叉验证，然后选出最优的。