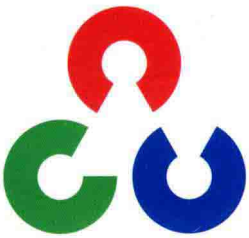
异步图书
www.epubit.com



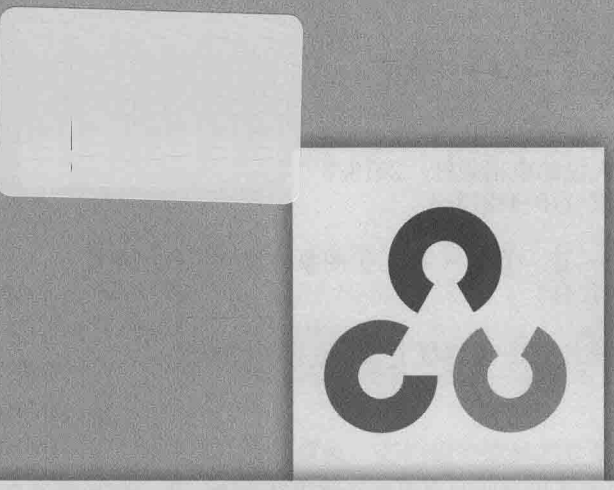
机器学习经典算法剖析

基于 OpenCV

赵春江◎编著

- 讲解了十大经典算法：正态贝叶斯分类器、K 近邻算法、支持向量机、决策树、AdaBoost、梯度提升树、随机森林、极端随机树、期望极大值和神经网络。
- 给出了算法相对应的 OpenCV 源码，并逐句详解。
- 给出了基于 OpenCV 的程序实现范例，充分体现了理论与实践相结合的特点。

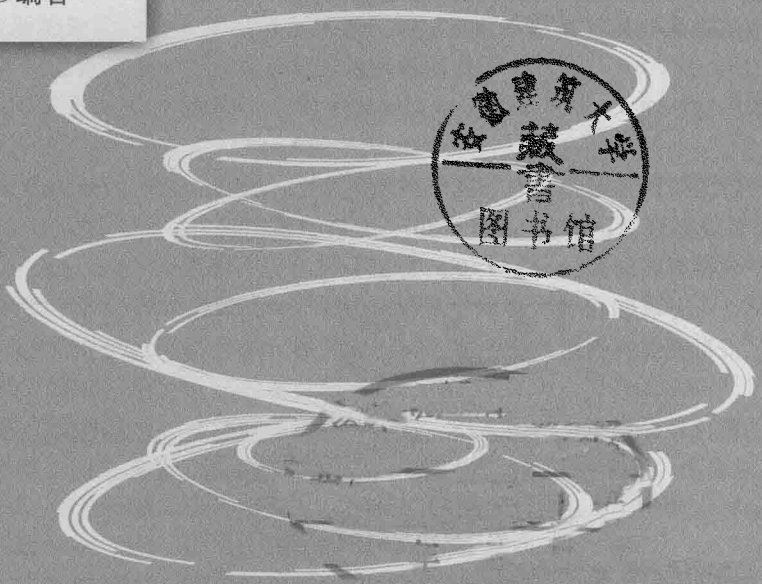
中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS



机器学习经典算法剖析

基于 OpenCV

赵春江◎编著



人民邮电出版社
北京

图书在版编目（CIP）数据

机器学习经典算法剖析 : 基于OpenCV / 赵春江编著
. -- 北京 : 人民邮电出版社, 2018.8
ISBN 978-7-115-48213-6

Ⅰ. ①机… Ⅱ. ①赵… Ⅲ. ①图象处理软件－程序设计 Ⅳ. ①TP391.413

中国版本图书馆CIP数据核字(2018)第064996号

内 容 提 要

机器学习是一种自动分析所构建模型的数据分析方法。通过迭代地从数据中不断学习，机器学习可以使计算机找到一些隐含的信息量，而这些信息量是无法明确通过编程得到的。

本书以 OpenCV 2.4.9 为研究工具，对机器学习经典算法——正态贝叶斯分类器、K 近邻算法、支持向量机、决策树、AdaBoost、梯度提升树、随机森林、极端随机树、期望极大值、神经网络等进行讲解，不仅具体分析了它们的原理和实现方法，还进行了详细的源码解析，并且给出了基于 OpenCV 的程序实现范例，充分体现了理论与实践相结合的特点。

本书适合于机器学习领域的工程技术人员阅读，也可供高等院校相关专业师生参考。

◆ 编　　著　赵春江
责任编辑　张　爽
责任印制　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
固安县铭成印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：18.75
字数：490 千字　　2018 年 8 月第 1 版
印数：1－2 400 册　　2018 年 8 月河北第 1 次印刷

定价：69.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

前　言

机器学习通常指的是能够结合人工智能完成各项任务的系统的变化过程。这些任务包括识别、诊断、规划、自动控制、预测等。而“变化”可以增强已有系统的性能，或者重新合成一个新的、更好的系统。

为什么机器必须要学习？或者为什么不从一开始就根据需要让机器完成任务？只有认识到机器学习的重要性，才能回答这两个问题。

❑ 有些任务只有通过实例的描述才能定义。也就是说，我们也许能够指定输入/输出对，但不能确定输入和它的期望输出之间的简要关系，所以我们就希望机器能够通过调整它的内部结构，从大量数据样本中产生正确的输出，并且这种输入/输出关系还能够反映那些隐含在样本中的信息。

❑ 隐藏在大量数据中的信息具有非常重要的关联性，机器学习方法能够提取出这些关联（数据挖掘）。

❑ 人类设计的机器往往在它们的工作环境下不能按照人类期望的那样很好地完成任务，部分原因可能是在设计之初，工作环境的某些特性不被人类所获知，机器学习方法就可以在工作中提高现有机器设计的性能。

❑ 关于某些任务，可用的知识量对于人类的显式编码来说可能太大了，而能够渐进地学习这些知识的机器可以获得比人类赋予它的更多的知识内容。

❑ 工作环境时刻都在改变，能够适应这种环境变化的机器可以减少不断重新设计的需要。

❑ 人类也在不断获取新的知识，为了使机器也掌握这些知识而不断重新设计系统是不现实的，而机器学习方法可以使机器像人类一样也能够跟踪这些新知识。

作为近 30 年兴起的一门交叉学科，机器学习已经成功地应用于许多领域，人们对它的需求也在不断增长。

OpenCV（Open Source Computer Vision）是一个主要应用于实时计算机视觉领域的程序函数库。OpenCV 是以 BSD 许可证授权发行的，程序源码完全对用户开放，它既可以直接免费使用，又可以作为二次开发的工具。OpenCV 函数库不仅包括了超过 500 个优化的图像和视频分析处理的算法，还包括了目前经典的十大机器学习算法——正态贝叶斯分类器、K 近邻算法、支持向量机、决策树、AdaBoost、梯度提升树、随机森林、极端随机树、期望极大值和神经网络。用户可以很方便地调用这些机器学习算法，用于自己的项目开发和科学研究。

对算法学习和理解的最高境界就是能够用编程语言实现它。但实现算法不仅要求对算法的原理一清二楚，还要求对编程语言熟练掌握，因此并不是每个人都能够达到这种境界。退而求其次，如果能够看懂别人写的程序，并且能够用该程序实现具体的应用，那么这也是一种学习算法的方法。OpenCV 这种开源的函数库正是我们学习算法的一个好的工具。

本书就是出于此目的，首先对正态贝叶斯分类器、K 近邻算法、支持向量机、决策树、AdaBoost、梯度提升树、随机森林、极端随机树、期望极大值、神经网络这十大经典的机器学习算法进行具体的原理分析，然后给出 OpenCV 的相关源码的逐句解释，最后完成一个基于 OpenCV 的应用实例。相信读者通过这 3 个步骤的学习，足以实现对算法理解的目的。

本书是作者利用业余时间撰写完成，加之作者水平有限，难免会有对算法和源码理解分析不到位的地方，权当抛砖引玉，恳请读者批评指正。

赵春江

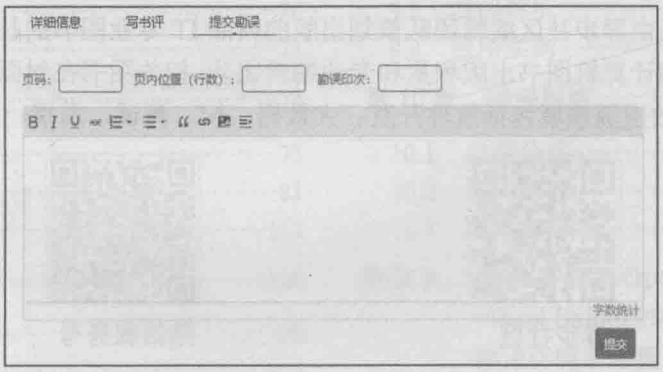
资源与支持

本书由异步社区出品，社区（https://www.epubit.com/）为您提供相关资源和后续服务。

提交勘误

作者和编辑尽最大努力来确保书中内容的准确性，但难免会存在疏漏。欢迎您将发现的问题反馈给我们，帮助我们提升图书的质量。

当您发现错误时，请登录异步社区，按书名搜索，进入本书页面，点击“提交勘误”，输入勘误信息，点击“提交”按钮即可。本书的作者和编辑会对您提交的勘误进行审核，确认并接受后，您将获赠异步社区的 100 积分。积分可用于在异步社区兑换优惠券、样书或奖品。



扫码关注本书

扫描下方二维码，您将会在异步社区微信服务号中看到本书信息及相关的服务提示。

与我们联系

我们的联系邮箱是 contact@epubit.com.cn。

如果您对本书有任何疑问或建议，请您发邮件给我们，并请在邮件标题中注明本书书名，

以便我们更高效地做出反馈。

如果您有兴趣出版图书、录制教学视频，或者参与图书翻译、技术审校等工作，可以发邮件给我们；有意出版图书的作者也可以到异步社区在线提交投稿（直接访问 www.epubit.com/selfpublish/submission 即可）。

如果您是学校、培训机构或企业，想批量购买本书或异步社区出版的其他图书，也可以发邮件给我们。

如果您在网上发现有针对异步社区出品图书的各种形式的盗版行为，包括对图书全部或部分内容的非授权传播，请您将怀疑有侵权行为的链接发邮件给我们。您的这一举动是对作者权益的保护，也是我们持续为您提供有价值的内容的动力之源。

关于异步社区和异步图书

“**异步社区**”是人民邮电出版社旗下 IT 专业图书社区，致力于出版精品 IT 技术图书和相关学习产品，为作译者提供优质出版服务。异步社区创办于 2015 年 8 月，提供大量精品 IT 技术图书和电子书，以及高品质技术文章和视频课程。更多详情请访问异步社区官网 https://www.epubit.com。

“**异步图书**”是由异步社区编辑团队策划出版的精品 IT 专业图书的品牌，依托于人民邮电出版社近 30 年的计算机图书出版积累和专业编辑团队，相关图书在封面上印有异步图书的 LOGO。异步图书的出版领域包括软件开发、大数据、AI、测试、前端、网络技术等。

异步社区

微信服务号

目　录

第1章 正态贝叶斯分类器

1.1 原理分析

OpenCV 实现的贝叶斯分类器不是朴素贝叶斯分类器（Naïve Bayes Classifier），而是正态贝叶斯分类器（Normal Bayes Classifier），两者虽然英文名称很相似，但它们是不同的贝叶斯分类器。前者在使用上有一个限制条件，那就是变量的特征之间要相互独立，而后者没有这个苛刻的条件，因此它的适用范围更广。为了保持理论的系统性和完整性，我们先介绍朴素贝叶斯分类器，然后再介绍正态贝叶斯分类器。

1. 朴素贝叶斯分类器

朴素贝叶斯分类器是一种基于贝叶斯理论的简单的概率分类器，而朴素的含义是指输入变量的特征属性之间具有很强的独立性。尽管这种“朴素”的设计和假设过于简单，但朴素贝叶斯分类器在许多复杂的实际情况下具有很好的表现，并且在综合性能上，该分类器要优于提升树（boosted trees）和随机森林（random forests）。

在许多实际应用中，对于朴素贝叶斯模型的参数估计往往使用的是极大似然法，因此可以这么认为，在不接受贝叶斯概率或不使用任何贝叶斯方法的前提下，我们仍然可以应用朴素贝叶斯模型对事物进行分类。

朴素贝叶斯分类器特别适用于输入变量的维数很高的情况，并且它只需要极少量的训练数据，就可以估计出分类所需的参数。

抽象地说，朴素贝叶斯是一种条件概率模型：要对一个个体进行分类，该个体用代表 n 个特征属性（相互独立的变量）的 n 维向量表示，即 $\boldsymbol{x}=(x_1,\cdots,x_n)^{\mathrm{T}}$，则分配给该个体的概率为：

$$p(C_k \mid x_1,\cdots,x_n) \tag{1-1}$$

式（1-1）表示 K 个可能输出或分类中第 k 个分类的概率，C_k 表示第 k 个响应输出，即分类结果。

如果个体的特征数量 n 很大，或者某个特征有大量的数值，则应用式（1-1）对个体进行分类不可行。因此，应用贝叶斯理论，把条件概率进行分解，使其更利于操作。

$$p(C_k|\boldsymbol{x})=\frac{p(C_k)p(\boldsymbol{x}|C_k)}{p(\boldsymbol{x})} \tag{1-2}$$

基于认识论的解释，概率是一种置信程度的度量。贝叶斯理论把某个事件在考虑证据之前和之后的置信程度关联了起来。回到式（1-2），$p(C_k)$表示在不考虑个体 $\boldsymbol{x}$ 的情况下，第 k 个分类的概率，我们把它定义为先验概率，而 $p(C_k|\boldsymbol{x})$表示在考虑个体 $\boldsymbol{x}$ 的情况下，第 k 个分类的概率，我们把它定义为后验概率，$p(\boldsymbol{x}|C_k)$定义为似然度，$p(\boldsymbol{x})$定义为标准化常量。

在实际应用中，我们仅关心式（1-2）中的分子部分，这是因为分母部分不依赖于分类结果 C，并且个体的特征属性 F_i 是给定的，所以分母是一个常数。

再来看式（1-2）中的分子部分，它是联合概率模型 $p(C_k, x_1, \cdots, x_n)$。基于链式法则，并重复应用条件概率的定义，这个联合概率模型可以重写为：

$$\begin{aligned} p(C_k, x_1, \cdots, x_n) &= p(C_k)p(x_1, \cdots, x_n \mid C_k) \\ &= p(C_k)p(x_1 \mid C_k)p(x_2, \cdots, x_n \mid C_k, x_1) \\ &= p(C_k)p(x_1 \mid C_k)p(x_2 \mid C_k, x_1)p(x_3, \cdots, x_n \mid C_k, x_1, x_2) \\ &= p(C_k)p(x_1 \mid C_k)p(x_2 \mid C_k, x_1) \quad p(x_n \mid C_k, x_1, x_2, x_3, \cdots, x_{n-1}) \end{aligned} \tag{1-3}$$

由于是“朴素”的贝叶斯，对于分类 C_k 来说，特征属性 F_i 是有条件的独立于特征属性 F_j 的，其中 $i \neq j$。因此，这意味着 $p(x_i \mid C_k, x_j) = p(x_i \mid C_k)$、$p(x_i \mid C_k, x_j, x_q) = p(x_i \mid C_k)$、$p(x_i \mid C_k, x_j, x_q, x_l) = p(x_i \mid C_k)$，依次类推，其中 $i \neq j, q, l$。则式（1-2）又可重写为：

$$p(C_k \mid x_1, \cdots, x_n) \propto p(C_k, x_1, \cdots, x_n)$$

$$\propto p(C_k)p(x_1 \mid C_k)p(x_2 \mid C_k)p(x_3 \mid C_k) \quad \propto p(C_k)\prod_{i=1}^{n} p(x_i \mid C_k) \tag{1-4}$$

到目前为止，我们得到了特征属性相互独立的朴素贝叶斯概率模型。利用该模型就可以得到具备决策规则的朴素贝叶斯分类器。而应用得最普遍的决策规则是最大后验概率（MAP），即选择最可能的假设，则朴素贝叶斯分类器所指定的分类结果为：

$$\hat{y} = \underset{k \in \{1, \cdots, K\}}{\operatorname{argmax}} \; p(C_k)\prod_{i=1}^{n} p(x_i \mid C_k) \tag{1-5}$$

训练朴素贝叶斯分类器的任务是估计两组参数：先验概率 $p(C_k)$和条件概率 $p(x_i \mid C_k)$。

先来计算条件概率 $p(x_i \mid C_k)$，它分为两种情况：一种是样本数据都是离散的形式，即样本的特征属性是离散的形式；另一种是样本数据都是连续的数值形式，即样本的特征属性是数值的形式。

当样本的特征属性是离散的形式时，条件概率 $p(x_i \mid C_k)$的估计为：

$$p(x_i = a_{il} \mid y = C_k) = \frac{\#D\{x_i = a_{il} \wedge y = C_k\}}{\#D\{y = C_k\}}, \qquad l = 1, 2, \cdots, S_i \tag{1-6}$$

式中，a_{il} 表示第 i 个特征属性可能取的第 l 个值；S_i 表示第 i 个特征属性可能选取的所有值的数量；$\#D\{X\}$表示在由 N 个训练样本构成的集合 D 中，满足条件 X 的样本的数量，因此分式中分母的含义是响应值为 C_k 的样本数，分子的含义是样本具有 a_{il} 值，并且响应值为 C_k 的数量。

式（1-6）给出了特征属性为 a_{il} 并且响应值为 C_k 的条件概率估计，该方法称为极大似然估计。但该方法可能会出现所要估计的概率值为 0 的情况，这时会影响到后验概率的计算结果，使分类产生偏差。采用平滑估计可以解决这个问题，即增加一个平滑系数 λ，则条件概率为：

$$p(x_i = a_{il} \mid y = C_k) = \frac{\#D\{x_i = a_{il} \wedge y = C_k\} + \lambda}{\#D\{y = C_k\} + S_i\lambda} \tag{1-7}$$

式中，$\lambda \geqslant 0$。显然，$\lambda = 0$ 为极大似然估计，当 $\lambda = 1$ 时，该平滑方法又称为拉普拉斯平滑。

当样本的特征属性是数值的形式时，条件概率的分布可以被认为是高斯分布，多项式分布或

伯努利分布，则它们的朴素贝叶斯分别被称为高斯朴素贝叶斯，多项式朴素贝叶斯和伯努利朴素贝叶斯。这里只介绍高斯朴素贝叶斯方法。

高斯朴素贝叶斯方法是假设对每一个可能的响应值 C_k，特征属性 x_i 是满足高斯正态分布的，即

$$p(x_i|y=C_k)=\frac{1}{\sqrt{2\pi\sigma_{ik}^2}}\mathrm{e}^{\frac{(x_i-\mu_{ik})^2}{2\sigma_{ik}^2}} \tag{1-8}$$

因此，必须估计出该高斯分布的均值 μ_{ik} 和方差 σ_{ik}^2：

$$\mu_{ik}=E[x_i\,|\,y=C_k] \tag{1-9}$$

$$\sigma_{ik}^2=E[(x_i-\mu_{ik})^2\,|\,y=C_k] \tag{1-10}$$

这里一共有 $2nK$ 个参数，这些参数都需要独立地去估计。估计的方法仍然可以采用极大似然估计。均值 μ_{ik} 的极大似然估计为：

$$\hat{\mu}_{ik}=\frac{1}{\sum_{j=1}^{N}\delta\left(y^{(j)}=C_k\right)}\sum_{j=1}^{N}x_i^{(j)}\delta\left(y^{(j)}=C_k\right) \tag{1-11}$$

式（1-11）中，上标 j 表示全部 N 个训练样本中的第 j 个样本，函数 $\delta(y=C_k)$ 表示：

$$\delta\left(y=C_k\right)=\begin{cases}1 & y=C_k\\ 0 & y\neq C_k\end{cases} \tag{1-12}$$

函数 δ 的作用就是选择那些响应值为 C_k 的训练样本。

方差 σ_{ik}^2 的极大似然估计为：

$$\hat{\sigma}_{ik}^2=\frac{1}{\sum_{j=1}^{N}\delta\left(y^{(j)}=C_k\right)}\sum_{j=1}^{N}\left(x_i^{(j)}-\hat{\mu}_{ik}\right)^2\delta\left(y^{(j)}=C_k\right) \tag{1-13}$$

采用极大似然估计得到的方差是有偏估计，因此往往采用最小方差无偏估计（MVUE）来取代极大似然估计，则此时的 σ_{ik}^2 估计为：

$$\hat{\sigma}_{ik}^2=\frac{1}{\left(\sum_{j=1}^{N}\delta\left(y^{(j)}=C_k\right)\right)-1}\sum_{j=1}^{N}\left(x_i^{(j)}-\hat{\mu}_{ik}\right)^2\delta\left(y^{(j)}=C_k\right) \tag{1-14}$$

下面再来讨论先验概率。贝叶斯分类器只能处理分类问题，即分类结果 C 具有离散的 K 个值，因此先验概率 $p(C_k)$ 的估计相对较简单。当已知所有的分类结果出现的概率都相等（如投骰子），则先验概率 $p(C_k)$ 为：

$$p\left(C_k\right)=\frac{1}{K} \tag{1-15}$$

当仅考虑训练样本数据时，则先验概率 $p(C_k)$ 为：

$$p\left(C_k\right)=\frac{N_k}{N} \tag{1-16}$$

式（1-15）中，N 表示训练样本的数量；N_k 表示分类结果为 C_k 的训练样本数量。式（1-16）这种极大似然估计仍然会有所要估计的概率值为 0 的情况，因此类似于式（1-7）改用平滑估计，得到先验概率为：

$$p(C_k)=\frac{N_k+\lambda}{N+K\lambda} \tag{1-17}$$

如果应用高斯朴素贝叶斯分类器预测样本，则先根据训练样本计算各个分类的先验概率，以及均值和方差，这样就得到了不同分类下的不同特征属性的高斯函数（见式（1-8））。然后把预测样本数据代入这些不同的高斯函数中，得到不同分类的各个特征属性的似然度。最后把同一分类的先验概率和不同特征属性的似然度相乘（见式（1-5）），哪个值大，该预测样本就属于该乘积所对应的分类。

2. 正态贝叶斯分类器

下面介绍正态贝叶斯分类器。该分类器只能处理特征属性是连续数值的分类问题。

正态贝叶斯分类器认为每一个分类的所有特征属性（即特征向量）服从多变量正态高斯分布，即：

$$p(x_1,\cdots,x_n|C_k)=\frac{1}{\sqrt{(2\pi)^n\left|\sum_k\right|}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu_k)^{\mathrm{T}}\sum_k^{-1}(\boldsymbol{x}-\mu_k)\right) \tag{1-18}$$

式中，μ_k表示第 k 个分类对应的 n 维均值向量；$|\sum_k|$表示第 k 个分类对应的 $n\times n$ 的协方差矩阵 $\sum_k$的行列式的值。

因此，该分类器认为特征属性之间不必是独立的，这比朴素贝叶斯的适用条件要宽。最终包括所有分类的整个分布函数是一个混合高斯分布，而每一个分类就是一个组件（component）。

由贝叶斯规则（见式（1-2））可知，后验概率正比于先验概率与似然度的乘积，但有些情况可以不考虑先验概率，如分类很少或维数较高等情况，即后验概率仅与似然度成正比，这样最大后验问题就变为了极大似然问题，这时只需要得到各个分类的似然度函数（见式（1-18）），代入新的预测样本，哪个值大，该样本就属于该似然度函数对应的分类，即

$$\hat{y}=\underset{k\in\{1,\cdots,K\}}{\operatorname{argmax}}\ p(x_1,\cdots,x_n\,|\,C_k) \tag{1-19}$$

有时为了计算方法，可以把式（1-18）取对数，成为对数似然函数，即：

$$\begin{aligned}\ln(L_k)&=-\frac{1}{2}\ln\left(\left|\sum_k\right|\right)-\frac{1}{2}(\boldsymbol{x}-\mu_k)^{\mathrm{T}}\sum_k^{-1}(\boldsymbol{x}-\mu_k)-\frac{n}{2}\ln(2\pi)\\&=-\frac{1}{2}\left[\ln\left(\left|\sum_k\right|\right)+(\boldsymbol{x}-\mu_k)^{\mathrm{T}}\sum_k^{-1}(\boldsymbol{x}-\mu_k)+n\ln(2\pi)\right]\end{aligned} \tag{1-20}$$

显然，求式（1-20）极大值问题可以转换为求式（1-20）中方括号内的极小值问题，其中最后一项 $n\ln(2\pi)$是常数，可以不用计算。

为了计算式（1-20），需要估计两组参数：均值向量 μ_k和协方差矩阵$\sum_k$。

均值向量 μ_k的第 i 个元素（即第 i 个特征属性）μ_{ki}的极大似然估计为：

$$\hat{\mu}_{ki}=\frac{\sum_{j=1}^{N_k}x_{ki}^{(j)}}{N_k} \tag{1-21}$$

式中，$x_{ki}^{(j)}$表示训练样本中属于分类 k 的第 j 个样本的第 i 个特征属性的值，则最终组成的 n 维（共有 n 个特征属性）均值向量 μ_k的极大似然估计为：

$$\hat{\mu}_k=(\hat{\mu}_{k1},\hat{\mu}_{k2},\cdots,\hat{\mu}_{kn})^{\mathrm{T}} \tag{1-22}$$

$n\times n$ 的协方差矩阵$\sum_k$的无偏估计形式为：

$$\widehat{\sum}_k = \frac{1}{N_k - 1}\begin{bmatrix} cov_k^{(1,1)} & cov_k^{(1,2)} & \cdots & cov_k^{(1,n)} \\ cov_k^{(2,1)} & cov_k^{(2,2)} & \cdots & cov_k^{(2,n)} \\ \vdots & \vdots & & \vdots \\ cov_k^{(n,1)} & cov_k^{(n,2)} & \cdots & cov_k^{(n,n)} \end{bmatrix} \tag{1-23}$$

其中，第 p 行第 q 列元素 $cov_k^{(p,q)}$ 表示训练样本中第 k 个分类所组成的数据集合中，第 p 个特征属性与第 q 个特征属性的协方差，如果 p 等于 q，则为方差。$cov_k^{(p,q)}$ 的具体形式为：

$$\begin{aligned} cov_k^{(p,q)} &= \sum_{j=1}^{N_k}\left[\left(x_{kp}^{(j)} - \hat{\mu}_{kp}\right)\left(x_{kq}^{(j)} - \hat{\mu}_{kq}\right)\right] \\ &= \sum_{j=1}^{N_k}\left(x_{kp}^{(j)}x_{kq}^{(j)} - x_{kp}^{(j)}\hat{\mu}_{kq} - x_{kq}^{(j)}\hat{\mu}_{kp} + \hat{\mu}_{kp}\hat{\mu}_{kq}\right) \\ &= \sum_{j=1}^{N_k}\left(x_{kp}^{(j)}x_{kq}^{(j)}\right) - \hat{\mu}_{kq}\sum_{j=1}^{N_k}x_{kp}^{(j)} - \hat{\mu}_{kp}\sum_{j=1}^{N_k}x_{kq}^{(j)} + N_k\hat{\mu}_{kp}\hat{\mu}_{kq} \end{aligned} \tag{1-24}$$

下面总结一下正态贝叶斯分类器的执行步骤。首先，由训练样本数据估计每个分类的协方差矩阵（式（1-23））和均值向量（式（1-22））；然后，把这两组变量代入式（1-20）中，从而得到每个分类的完整的对数似然函数。当需要预测样本时，把样本的特征属性分别代入全部分类的对数似然函数中，最大对数似然函数对应的分类就是该样本的分类结果。

下面举一个例子，该例子是维基百科中英文条目 Naive Bayes classifier 列举的例子。

表 1-1 是某国人体特征指标的一组统计数据。

表 1-1　　人体特征指标数据

序号	性别	身高（英尺）	体重（磅）	脚掌长（英寸）
1	男	6	180	12
2	男	5.92	190	11
3	男	5.58	170	12
4	男	5.92	165	10
5	女	5	100	6
6	女	5.5	150	8
7	女	5.42	130	7
8	女	5.75	150	9

其中，1 英尺=0.3048m，1 磅=0.453592kg，1 英寸=0.0254m。

表 1-1 中的样本一共有 8 个，分为男和女两类，并各有样本数 4 个，即 $N_{男}=4$，$N_{女}=4$。

由式（1-21）先计算男人的均值向量 $\mu_{男}$：

$$\hat{\mu}_{男身高} = \frac{6+5.92+5.58+5.92}{N_{男}} = \frac{23.42}{4} = 5.855$$

$$\hat{\mu}_{男体重} = \frac{180+190+170+165}{N_{男}} = \frac{705}{4} = 176.25$$

$$\hat{\mu}_{男脚掌长}=\frac{12+11+12+10}{N_{男}}=\frac{45}{4}=11.25$$

则：

$$\hat{\mu}_{男}=\left(\hat{\mu}_{男身高},\hat{\mu}_{男体重},\hat{\mu}_{男脚掌长}\right)^{\mathrm{T}}=(5.855,176.25,11.25)^{\mathrm{T}}$$

再计算女人的均值向量 $\mu_{女}$：

$$\hat{\mu}_{女身高}=\frac{5+5.5+5.42+5.75}{N_{女}}=\frac{21.67}{4}=5.4175$$

$$\hat{\mu}_{女体重}=\frac{100+150+130+150}{N_{女}}=\frac{530}{4}=132.5$$

$$\hat{\mu}_{女脚掌长}=\frac{6+8+7+9}{N_{女}}=\frac{30}{4}=7.5$$

则：

$$\hat{\mu}_{女}=\left(\hat{\mu}_{女身高},\hat{\mu}_{女体重},\hat{\mu}_{女脚掌长}\right)^{\mathrm{T}}=(5.4175,132.5,7.5)^{\mathrm{T}}$$

然后由式（1-23）和式（1-24）计算男人和女人的协方差矩阵，它们都是 3×3 的对称方阵。这里仅以男人的身高和体重为例，计算它们的协方差 $cov_{男}^{(身高,体重)}$：

$$\begin{aligned}cov_{男}^{(身高,体重)}&=(6-5.855)\times(180-176.25)+(5.92-5.855)\times(190-176.25)\\&\quad+(5.58-5.855)\times(170-176.25)+(5.92-5.855)\times(165-176.25)\\&=0.5437+0.8937+1.7188-0.7312=2.425\end{aligned}$$

则最终男人和女人的协方差矩阵分别为：

$$\sum_{男}=\begin{bmatrix}0.035 & 0.808 & -0.065\\ 0.808 & 122.917 & 2.917\\ -0.065 & 2.917 & 0.917\end{bmatrix}\qquad \sum_{女}=\begin{bmatrix}0.097 & 6.942 & 0.388\\ 6.942 & 558.333 & 28.333\\ 0.388 & 28.333 & 1.667\end{bmatrix}$$

把得到的男人和女人的均值向量和协方差矩阵代入式（1-20）中，就得到两个分类的对数似然函数。由于该函数需要的是协方差矩阵的逆矩阵和行列式的值，所以还需要计算这两组值：

$$\left|\sum_{男}\right|=2.225\qquad \left|\sum_{女}\right|=0.669$$

$$\sum_{男}^{-1}=\begin{bmatrix}46.826 & -0.418 & 4.651\\ -0.418 & 0.013 & -0.070\\ 4.651 & -0.070 & 1.642\end{bmatrix}\qquad \sum_{女}^{-1}=\begin{bmatrix}191.004 & -0.847 & -30.104\\ -0.847 & 0.017 & -0.089\\ -30.104 & -0.089 & 9.114\end{bmatrix}$$

则对数似然函数为：

$$\ln\left(L_{男}\right)=-\frac{1}{2}\left[\ln 2.225+\left(\boldsymbol{x}-\begin{bmatrix}5.855\\ 176.25\\ 11.25\end{bmatrix}\right)^{\mathrm{T}}\begin{bmatrix}46.826 & -0.418 & 4.651\\ -0.418 & 0.013 & -0.070\\ 4.651 & -0.070 & 1.642\end{bmatrix}\left(\boldsymbol{x}-\begin{bmatrix}5.855\\ 176.25\\ 11.25\end{bmatrix}\right)+3\ln(2\pi)\right]$$

$$\ln\left(L_{女}\right)=-\frac{1}{2}\left[\ln 0.669+\left(\boldsymbol{x}-\begin{bmatrix}5.4175\\ 132.5\\ 7.5\end{bmatrix}\right)^{\mathrm{T}}\begin{bmatrix}191.004 & -0.847 & -30.104\\ -0.847 & 0.017 & -0.089\\ -30.104 & -0.089 & 9.114\end{bmatrix}\left(\boldsymbol{x}-\begin{bmatrix}5.4175\\ 132.5\\ 7.5\end{bmatrix}\right)+3\ln(2\pi)\right]$$

已知某人身高 6 英尺，体重 130 磅，脚掌长 8 英寸，利用前面的训练样本预测该人是男，还是女。预测样本向量 $\boldsymbol{x}=(6, 130, 8)^{\mathrm{T}}$，代入 $\ln(L_{男})$和 $\ln(L_{女})$，则分别为−16.314 和−28.730。显然，$\ln(L_{男})$大于 $\ln(L_{女})$，所以这个人可能是男人。

这里要说明的是，维基百科中应用的是高斯朴素贝叶斯分类器，即前面第一部分介绍的内容，它得到的结论是该人为女人。从中可以看出，高斯朴素贝叶斯分类器与正态贝叶斯分类器不同，正态贝叶斯分类器不在乎特征属性之间是否相互独立，本人认为正态贝叶斯分类器应该更准确一些。

实际计算似然度函数时，应用奇异值分解会使程序更简洁。由于协方差矩阵$\sum_k$是对称矩阵，所以它的奇异值分解为：

$$\sum_k = \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^{\mathrm{T}} = \boldsymbol{U}\boldsymbol{W}\boldsymbol{U}^{\mathrm{T}} \tag{1-25}$$

式中，$\boldsymbol{W}$ 是由特征值组成的对角线矩阵；$\boldsymbol{U}$ 是由特征向量组成的正交矩阵，具有 $\boldsymbol{U}^{-1}=\boldsymbol{U}^{\mathrm{T}}$ 的性质，则$\sum_k$的逆矩阵为：

$$\sum_k^{-1} = \left(\boldsymbol{U}\boldsymbol{W}\boldsymbol{U}^{\mathrm{T}}\right)^{-1} = \left(\boldsymbol{U}^{\mathrm{T}}\right)^{-1}\left(\boldsymbol{U}\boldsymbol{W}\right)^{-1} = \left(\boldsymbol{U}^{-1}\right)^{-1}\boldsymbol{W}^{-1}\boldsymbol{U}^{-1} = \boldsymbol{U}\boldsymbol{W}^{-1}\boldsymbol{U}^{\mathrm{T}} \tag{1-26}$$

设 $\boldsymbol{D}=\boldsymbol{x}-\mu_k$，则$(\boldsymbol{x}-\mu_k)^{\mathrm{T}}\sum_k^{-1}(\boldsymbol{x}-\mu_k)$为

$$\left(\boldsymbol{x}-\mu_k\right)^{\mathrm{T}}\sum_k^{-1}\left(\boldsymbol{x}-\mu_k\right) = \boldsymbol{D}^{\mathrm{T}}\boldsymbol{U}\boldsymbol{W}^{-1}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{D} = \left(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}\right)\boldsymbol{W}^{-1}\left(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{D}\right) = \left(\boldsymbol{D}^{\mathrm{T}}\boldsymbol{U}\right)\boldsymbol{W}^{-1}\left(\boldsymbol{D}^{\mathrm{T}}\boldsymbol{U}\right)^{\mathrm{T}} \tag{1-27}$$

式中，$\boldsymbol{D}^{\mathrm{T}}$是行向量；$\boldsymbol{U}$ 是方阵，则 $\boldsymbol{D}^{\mathrm{T}}\boldsymbol{U}$ 为行向量，设该行向量的元素为 a_i，则$(\boldsymbol{D}^{\mathrm{T}}\boldsymbol{U})^{\mathrm{T}}$为列向量，因为 $\boldsymbol{W}$ 是对角线矩阵，设该矩阵对角线上的元素为 w_i，由矩阵的知识可知，$\boldsymbol{W}^{-1}$ 也是对角线矩阵，并且其对角线上的元素为 $1/w_i$，则式（1-27）改写为：

$$\begin{aligned}
\left(\boldsymbol{x}-\mu_k\right)^{\mathrm{T}}\sum_k^{-1}\left(\boldsymbol{x}-\mu_k\right) &= \left(\boldsymbol{D}^{\mathrm{T}}\boldsymbol{U}\right)W^{-1}\left(\boldsymbol{D}^{\mathrm{T}}\boldsymbol{U}\right)^{\mathrm{T}} \\
&= \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}\begin{bmatrix} w_1^{-1} & 0 & \cdots & 0 \\ 0 & w_2^{-1} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & w_n^{-1} \end{bmatrix}\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \\
&= \begin{bmatrix} a_1 w_1^{-1} & a_2 w_2^{-1} & \cdots & a_n w_n^{-1} \end{bmatrix}\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \\
&= \frac{a_1 a_1}{w_1} + \frac{a_2 a_2}{w_2} + \cdots + \frac{a_n a_n}{w_n} \\
&= \sum_{i=1}^{n}\frac{a_i a_i}{w_i}
\end{aligned} \tag{1-28}$$

我们还注意到一个性质，那就是行列式的值等于该矩阵特征值的乘积，前面我们已经得到$\sum_k$的特征值为 w_i，则它的行列式值为：

$$\left|\sum_k\right| = \prod_{i=1}^{n} w_i \tag{1-29}$$

1.2 源码解析

下面详细分析 OpenCV 中的贝叶斯分类器的源码。再次强调的是，OpenCV 实现的是正态贝叶斯分类器，不是朴素贝叶斯分类器。

CvNormalBayesClassifier 类的默认构造函数：

```
CvNormalBayesClassifier::CvNormalBayesClassifier()
{
    var_count = var_all = 0;
    var_idx = 0;
    cls_labels = 0;
    count = 0;
    sum = 0;
    productsum = 0;
    avg = 0;
    inv_eigen_values = 0;
    cov_rotate_mats = 0;
    c = 0;
    default_model_name = "my_nb";
}
```

构建正态贝叶斯分类器的函数：

```
bool CvNormalBayesClassifier::train( const CvMat* _train_data, const CvMat* _responses,
                                     const CvMat* _var_idx, const CvMat* _sample_idx, bool
update )
{
    const float min_variation = FLT_EPSILON;     //定义一个很小的数
    bool result = false;     //函数返回的标识变量
    CvMat* responses = 0;     //表示分类结果，即响应值
    const float** train_data = 0;     //表示训练样本数据
    CvMat* __cls_labels = 0;     //表示样本响应值的标签
    CvMat* __var_idx = 0;     //表示特征属性的索引
    CvMat* cov = 0;     //表示某个分类的协方差矩阵

    CV_FUNCNAME( "CvNormalBayesClassifier::train" );

    __BEGIN__;

    int cls, nsamples = 0, _var_count = 0, _var_all = 0, nclasses = 0;
    int s, c1, c2;
    const int* responses_data;     //指向响应值
    //*调用 cvPrepareTrainData 函数，首先判断输入参数_train_data 和_responses 是否正确，然后由
参数_var_idx 和_sample_idx 得到真正要训练的样本数据 train_data，由参数_sample_idx 得到所对应的响
应值 responses，nsamples 为得到的 train_data 中的样本数量，_var_count 为 train_data 中样本特征的
数量，_var_all 为_train_data 中每个样本应该有的特征的数量，__cls_labels 为响应值的分类标签的映射矩
阵，__var_idx 为由参数_var_idx 从_var_all 中提取的每个样本的特征的掩码矩阵*//
    CV_CALL( cvPrepareTrainData( 0,
        _train_data, CV_ROW_SAMPLE, _responses, CV_VAR_CATEGORICAL,
        _var_idx, _sample_idx, false, &train_data,
        &nsamples, &_var_count, &_var_all, &responses,
        &__cls_labels, &__var_idx ));

    if(!update )     //不更新数据，即由样本数据重新建立贝叶斯分类器
    {
        const size_t mat_size = sizeof(CvMat*);
        size_t data_size;

        clear();

        var_idx = __var_idx;     //样本特征的掩码矩阵
        cls_labels = __cls_labels;     //响应值的分类标签矩阵
        __var_idx = __cls_labels = 0;     //清零
        var_count = _var_count;     //真正的训练样本用到的特征数量
```

```
    var_all = _var_all;     //全部样本的特征数量

    nclasses = cls_labels->cols;     //表示分类的数量，即 K
    data_size = nclasses*6*mat_size;     //定义所需的全部数据的内存空间大小

    CV_CALL( count = (CvMat**)cvAlloc( data_size ));     //分配空间
    memset( count, 0, data_size );     //清零
    //定义不同的空间
    //count 表示每个分类的样本数量，即变量 N_k
    //sum 表示分类中的每种特征属性值的和，即式（1-21）的分子部分
    sum             = count      + nclasses;
    //productsum 表示式（1-24）中的第一个∑
    productsum      = sum        + nclasses;
    //avg 表示均值向量，即式（1-21）
    avg             = productsum + nclasses;
    //inv_eigen_values 表示每个分类的协方差矩阵∑_k 的特征值，最终存储的是特征值的倒数，即式
    （1-28）中的 1/w_i
    inv_eigen_values= avg        + nclasses;
    //cov_rotate_mats 表示每个分类的协方差矩阵的特征向量矩阵的转置，即式（1-25）中的 U^T
    cov_rotate_mats = inv_eigen_values          + nclasses;
    //创建矩阵 c，用来表示式（1-20）中的 ln(|∑_k|)
    CV_CALL( c = cvCreateMat( 1, nclasses, CV_64FC1 ));
    //遍历所有分类，创建上面 6 个矩阵，并清零
    for( cls = 0; cls < nclasses; cls++ )
    {
        // count 矩阵的大小为 K×n
        CV_CALL(count[cls]        = cvCreateMat( 1, var_count, CV_32SC1 ));
        // sum 矩阵的大小为 K×n
        CV_CALL(sum[cls]          = cvCreateMat( 1, var_count, CV_64FC1 ));
        // productsum 矩阵的大小为 K×n×n
        CV_CALL(productsum[cls]   = cvCreateMat( var_count, var_count, CV_64FC1 ));
        // avg 矩阵的大小为 K×n
        CV_CALL(avg[cls]              = cvCreateMat( 1, var_count, CV_64FC1 ));
        // inv_eigen_values 矩阵的大小为 K×n
        CV_CALL(inv_eigen_values[cls] = cvCreateMat( 1, var_count, CV_64FC1 ));
        // cov_rotate_mats 矩阵的大小为 K×n×n
        CV_CALL(cov_rotate_mats[cls]  = cvCreateMat( var_count, var_count, CV_64FC1 ));
        CV_CALL(cvZero( count[cls] ));
        CV_CALL(cvZero( sum[cls] ));
        CV_CALL(cvZero( productsum[cls] ));
        CV_CALL(cvZero( avg[cls] ));
        CV_CALL(cvZero( inv_eigen_values[cls] ));
        CV_CALL(cvZero( cov_rotate_mats[cls] ));
    }
}
else     //在已有的贝叶斯分类器的基础上，添加新的训练样本
{
    // check that the new training data has the same dimensionality etc.
    if( _var_count != var_count || _var_all != var_all || !((!_var_idx && !var_idx) ||
        (_var_idx && var_idx && cvNorm(_var_idx,var_idx,CV_C) < DBL_EPSILON)) )
        CV_ERROR( CV_StsBadArg,
        "The new training data is inconsistent with the original training data" );

    if( cls_labels->cols != __cls_labels->cols ||
        cvNorm(cls_labels, __cls_labels, CV_C) > DBL_EPSILON )
        CV_ERROR( CV_StsNotImplemented,
        "In the current implementation the new training data must have absolutely"
        "the same set of class labels as used in the original training data" );

    nclasses = cls_labels->cols;
}

responses_data = responses->data.i;     //指向训练样本的响应值矩阵
//创建 cov 矩阵，表示协方差矩阵
CV_CALL( cov = cvCreateMat( _var_count, _var_count, CV_64FC1 ));

/* process train data (count, sum , productsum) */
//遍历所有的训练样本，计算式（1-21）的分子和分母部分，以及式（1-24）中的第一个∑部分
for( s = 0; s < nsamples; s++ )
```

```
    {
        cls = responses_data[s];     //得到该训练样本的响应值，即分类
        //定义 3 个矩阵 count、sum 和 productsum 的指针
        int* count_data = count[cls]->data.i;
        double* sum_data = sum[cls]->data.db;
        double* prod_data = productsum[cls]->data.db;
        const float* train_vec = train_data[s];     //得到该训练样本数据
        //遍历所有特征
        for( c1 = 0; c1 < _var_count; c1++, prod_data += _var_count )
        {
            //得到该训练样本的第 c1 个特征属性的值，即式（1-21）中的 x_ki^(j)
            double val1 = train_vec[c1];
            sum_data[c1] += val1;     //计算式（1-21）中的分子部分
            count_data[c1]++;     //计算式（1-21）中的分母部分
            //计算式（1-24）中的第一个∑，即∑x_kp^(j) x_kq^(j)，该算式组成的矩阵是对称矩阵，所以只计算该对
            //称矩阵的一半即可，这里的 for 循环计算的是该矩阵的右上角部分
            for( c2 = c1; c2 < _var_count; c2++ )
                prod_data[c2] += train_vec[c2]*val1;
        }
    }
    cvReleaseMat( &responses );     //释放 responses 矩阵
    responses = 0;

    /* calculate avg, covariance matrix, c */
    //遍历所有响应值，即分类结果，计算式（1-21）和式（1-23），即均值向量和协方差矩阵，以及式（1-20）
    //中的 ln(|∑_k|)
    for( cls = 0; cls < nclasses; cls++ )
    {
        double det = 1;     //表示协方差矩阵∑_k 的行列式值
        int i, j;
        CvMat* w = inv_eigen_values[cls];     //表示协方差矩阵|∑_k|的特征值，即 w_i
        //定义 3 个矩阵 count、avg 和 sum 的指针
        int* count_data = count[cls]->data.i;
        double* avg_data = avg[cls]->data.db;
        double* sum1 = sum[cls]->data.db;
        //productsum 矩阵是对称矩阵，前面只计算了该矩阵的右上角部分，这里调用 cvCompleteSymm 函数，
        //完成最终的对称矩阵，即把右上角的数据对称复制到左下角
        cvCompleteSymm( productsum[cls], 0 );
        //遍历所有特征属性
        for( j = 0; j < _var_count; j++ )
        {
            int n = count_data[j];     //得到当前分类的第 j 个特征的数量，即 N_k
            avg_data[j] = n ? sum1[j] / n : 0.;     //得到均值向量，即式（1-21）
        }
        //指针重新指向矩阵的首地址
        count_data = count[cls]->data.i;
        avg_data = avg[cls]->data.db;
        sum1 = sum[cls]->data.db;

        for( i = 0; i < _var_count; i++ )     //遍历所有特征
        {
            double* avg2_data = avg[cls]->data.db;     //指向均值向量
            //该指针指向的变量表示的含义是式（1-24）中的∑x_kq^(j)
            double* sum2 = sum[cls]->data.db;
            //指向 productsum 矩阵，即式（1-24）中的∑x_kp^(j) x_kq^(j)
            double* prod_data = productsum[cls]->data.db + i*_var_count;
            double* cov_data = cov->data.db + i*_var_count;     //指向协方差矩阵
            double s1val = sum1[i];     //表示式（1-24）中的∑x_kp^(j)
            double avg1 = avg_data[i];     //表示式（1-24）中的 μ_kp 的估计
            //当前分类的第 i 个特征的数量，即式（1-24）中的 N_k
            int _count = count_data[i];
            //遍历前 i 个特征，即只计算了协方差矩阵（式（1-23））的左下角部分
            for( j = 0; j <= i; j++ )
            {
                double avg2 = avg2_data[j];     //表示式（1-24）中的 μ_kq 的估计
                //式（1-24）
                double cov_val = prod_data[j] - avg1 * sum2[j] - avg2 * s1val + avg1 *
                avg2 * _count;
                //得到协方差，即式（1-23）中的元素
```