第 5 章 决策树与随机森林

5.1 决策树

树模型的核心思想是将特征空间划分为若干互不重叠的区域，并且在每个区域内构建一个局部模型，用于对落入该区域的样本进行预测。确定样本所属区域的过程，是根据其特征取值进行一系列分层判断，这一过程可以直观地表示为树状结构。这里的“局部模型”指的是仅使用训练集中被划入该区域的样本来构建对应的预测模型。

假设我们需要根据当前的温度和湿度来决定活动是在室内还是户外进行。如图5.1(a) 所示，我们依据温度和湿度的不同取值，将特征空间划分成五个互不重叠的区域，记为 $\{R_1, \ldots, R_5\}$。对于每一个区域，我们给出相应的活动安排，即选择“室内”或“户外”。根据温度和湿度划分为五个不同区域的过程可以表示为如图5.1(b) 所示的决策树。具体过程如下：首先根据温度是否高于 25° 摄氏度分成低温和高温两部分。对于低温部分，再判断温度是否低于 0°，若温度低于或等于 0° 摄氏度，则活动安排在室内；若高于 0° 摄氏度，则进一步根据湿度进行判断，当湿度小于等于 75% 时安排在户外，否则安排在室内。对于高温部分，则直接根据湿度进行决择，当湿度小于等于 50% 时活动安排在户外，否则安排在室内。

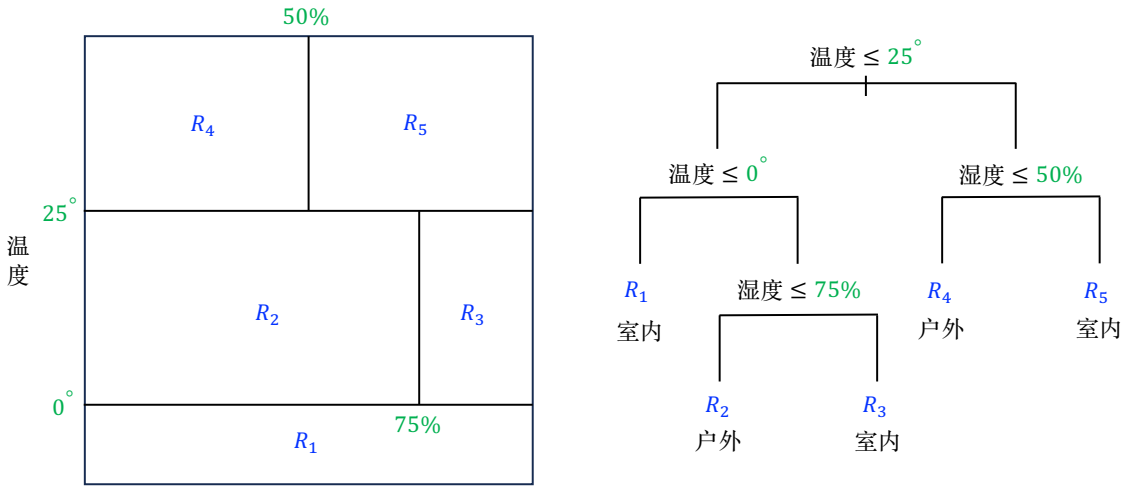


图 5.1: 树模型的特征空间划分示意图。(a) 根据温度和湿度的不同取值，将二维输入空间划分为五个互不重叠的区域 $\{R_1, \ldots, R_5\}$。(b) 对应于这些区域划分的决策树表示。

上述决策树模型可以形式化表示为：

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} c_m \mathbb{I}(\boldsymbol{x} \in R_m) \tag{5.1}$$

其中 R_m 表示第 m 个区域，c_m 表示对应区域的预测值，指示函数 $\mathbb{I}(\boldsymbol{x} \in R_m)$ 的取值非 1 即 0（其中 1 表示 $\boldsymbol{x}$ 属于该区域，0 则表示不属于）。在此分类的例子中，$M = 5$，而 c_m 为“室内”或“户外”所对应的取值。公式5.1的含义是：对于给定输入 $\boldsymbol{x}$，先确定其所属的区域 R_m，然后将该区域对应的取值 c_m 作为预测结果。对于分类问题，c_m 通常取训练集中落入区域 R_m 的样本中出现次数最多的类别；而对于回归问题，c_m 则通常取落入区域 R_m 的所有样本的均值。

决策树的一个主要优点在于其具备良好的可解释性。从树根节点到叶节点（即预测结果）的每一条路径都可以转化为一条明确的决策规则。以图5.1(b) 所示的决策树为例，我们可以将其化为如下五条规则，每条规则对应一个区域：

$$
\begin{aligned}
(\text{温度} \le 0^\circ) &\Rightarrow \text{室内活动} \\
(\text{温度} \le 25^\circ) \text{ 且 } (\text{温度} > 0^\circ) \text{ 且 } (\text{湿度} \le 75\%) &\Rightarrow \text{户外活动} \\
(\text{温度} \le 25^\circ) \text{ 且 } (\text{温度} > 0^\circ) \text{ 且 } (\text{湿度} > 75\%) &\Rightarrow \text{室内活动} \\
(\text{温度} > 25^\circ) \text{ 且 } (\text{湿度} \le 50\%) &\Rightarrow \text{户外活动} \\
(\text{温度} > 25^\circ) \text{ 且 } (\text{湿度} > 50\%) &\Rightarrow \text{室内活动}
\end{aligned}
$$

决策树的另一个显著优点是对一些极端异常值具有较强的鲁棒性。决策树通过递归地划分特征空间，将输入空间切分为图 5.1(a) 所示的一个个矩形区域。由于异常值通常具有与大多数样本不同的特征值，树模型往往能够通过少数次分割便将其“隔离”到某个特定的叶节点中。一旦异常值被隔离到某个小分支，它对预测的影响就会被局限在该小分支范围内，而不会波及树中其他分支。此外，决策树在选择分裂阈值时，主要依据的是特征值之间的相对顺序，而非它们的具体数值大小。因此，只要异常值在排序中的相对位置未发生改变，其极端数值通常不会显著影响分裂阈值的选择，从而也不会改变树的整体结构。

构建决策树的关键在于为每个节点选择合适的划分特征及其对应的分裂阈值。由于整棵树所有可能的划分组合构成庞大的搜索空间，直接一次性确定所有节点的特征与阈值几乎是不可行的。因此，决策树通常采用一种自顶向下、逐层扩展的贪婪策略。这种方法从根节点开始逐步扩展树结构，并在每个节点上选择当前能够最大程度降低损失函数的特征及其阈值，以实现最优的局部划分。在后续两节中，我们将分别介绍回归树和分类树中用于选择划分特征与确定分裂阈值的准则，以及它们所对应的损失函数形式。

5.2 回归树

我们先介绍用于回归任务的决策树构建方法。设包含 N 个样本的训练集 $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$。每个样本 $\boldsymbol{x}_i$ 含有 D 个特征，可表示为 $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{iD})$。算法需要自动为每个节点选择划分的特征及其阈值，从而确定整棵树的拓扑结构（形状）。对于回归树，常用的分裂准则是最小化平方误差损失：

$$
\mathcal{L}(R_m) = \frac{1}{|R_m|} \sum_{\boldsymbol{x}_i \in R_m} (y_i - f(\boldsymbol{x}_i))^2 \tag{5.2}
$$

其中 R_m 表示划分得到的某个区域，而 $|R_m|$ 表示该区域内的样本数量。容易看出，为了最小化上述平方误差，公式 5.2 中的 $f(\boldsymbol{x}_i)$ 应取该区域内所有样本的输出值的平均值，即：

$$f(\boldsymbol{x}_i) = c_m = \text{Ave}(y_i|\boldsymbol{x}_i \in R_m) \tag{5.3}$$

其中 Ave 表示取平均。

如前所述，我们采用贪婪策略来构建回归树。首先从整个训练集出发，考虑某个特征 j 及其分裂阈值 t，并根据该阈值将训练集划分为如下两部分：

$$R_1(j,t) = \{\boldsymbol{x}_i|x_{ij} \leq t\} \quad 和 \quad R_2(j,t) = \{\boldsymbol{x}_i|x_{ij} > t\} \tag{5.4}$$

为了选择最佳的划分特征 j 与阈值 t，我们最小化划分后两部分的平方误差和：

$$\min_{j,t}\left[\min_{c_1}\sum_{\boldsymbol{x}_i\in R_1(j,t)}(y_i-c_1)^2 + \min_{c_2}\sum_{\boldsymbol{x}_i\in R_2(j,t)}(y_i-c_2)^2\right] \tag{5.5}$$

对于选定的划分特征 j 及其阈值 t，上式内部的最小化可以通过使用公式5.3计算各区域内样本输出的均值来得到。

对于离散特征，其可能取的值是有限的。对于连续特征，虽然理论上可以取无限多的数值，但由于训练样本数量有限，实际出现的不同特征值也是有限的。对于连续型特征，可能的分裂阈值通常取相邻两个特征值的平均值。若训练集中共有 N 个样本，则每个连续特征可能的划分阈值最多为 $N-1$ 种。由于特征总数是有限的，每个特征可能的划分阈值也是有限的，这使得可以高效地搜索出使公式5.5最小的最优特征 j 及其阈值 t。

比较难处理的情况是某些离散特征的取值之间没有自然的顺序关系，即无法对其取值进行排序。例如颜色特征可能取 “红”、“绿”、“蓝” 等不同类别，但我们无法说 “红” 比 “蓝” 更大或更小，因此也不能简单地用类似颜色小于等于 “绿” 的规则将样本划分为两部分。对于具有 C 种不同取值的无序离散特征，其可能的划分方式有 $2^{C-1}-1$ 种[1]。当 C 较大时，直接枚举所有划分并最小化公式5.5的方法在计算上会变得不可行。对于回归问题，常用的处理方法是先计算每个离散取值对应的样本输出值的平均值，然后根据这些均值对取值进行排序，从而将原本无序的特征转换为相对于预测值有序的特征 [24]。

对于具有多种取值（至少三种）的离散特征，是否可以根据其不同取值将样本一次性划分为多个子集，而非每次都只分成两个子集？尤其是对于无序离散特征，多叉划分似乎可以避免在二叉划分中为寻找最优子集而额外进行的对于特征值的排序步骤。这个问题的回答是：当特征的取值数量不多时，确实可以考虑一次性的多叉划分，此时决策树的结构也会由二叉树扩展为多叉树。然而，这并不是普遍采用的做法，主要有以下两点原因。一是：一次划分成多个子集可能会过快地 “切碎” 样本集合，使得下一层节点包含的样本数量过少；二是：通过多次二叉划分，同样可以实现将样本分成多个组的效果。基于这两点考虑，实践中我们仍然倾向于采用每次仅将样本划分为两部分的二叉划分策略。

[1]假设划分为 R_1 和 R_2 两类，每种不同取值可以分配到 R_1 或 R_2，共有 2^C 种可能。排除 R_1 或 R_2 为空的两种情况，得到 2^C-2 种。由于不区分 (R_1,R_2) 与 (R_2,R_1) 这两种等价的划分，还需除以 2，最终得到 $2^{C-1}-1$ 种不同划分。

输入		输出
x_1	x_2	y
0	0	0
0	1	1
1	0	0
1	1	1

异或运算

$$\mathcal{L}(R) = \frac{1}{4}[(0-0.5)^2+(1-0.5)^2+(0-0.5)^2+(1-0.5)^2] = 0.25$$

选择特征x_1或x_2进行划分：

$$\mathcal{L}(R_1) = \frac{1}{2}[(0-0.5)^2+(1-0.5)^2] = 0.25$$

$$\mathcal{L}(R_2) = \frac{1}{2}[(0-0.5)^2+(1-0.5)^2] = 0.25$$

$$\Delta = \mathcal{L}(R) - \left(\frac{1}{2}\mathcal{L}(R_1) + \frac{1}{2}\mathcal{L}(R_2)\right) = 0.25 - \left(\frac{1}{2}\times 0.25 + \frac{1}{2}\times 0.25\right) = 0$$

图 5.2: 增益阈值作为停止条件时可能出现“短视”现象。左图显示异或运算输入与输出的关系：当两个输入 x_1 和 x_2 相同时，输出 y 为 0；当两个输入相异时，输出为 1。右图展示增益的计算过程，即不划分时的损失 $\mathcal{L}(R)$ 与划分后损失函数值的变化。计算结果表明，无论选择特征 x_1，还是 x_2 进行划分，增益均为零。然而，这种判断显然是“短视”的。因为随机选择特征 x_1 或 x_2 进行划分，再使用另一个特征进一步划分，则能够成功构建对应异或函数的决策树。

既然二分策略具有较好的普适性，并且每次划分都可以高效地找到最优的特征及其阈值，我们便可以据此逐步构建决策树。具体而言，从完整的训练集出发，并获得最佳分裂后，将样本划分到对应的两个子区域，再在每个新生成的区域内重复相同的划分步骤，直至满足预设的停止条件。需要注意的是，每次划分仅作用于当前区域内的样本，也就是说，进一步的分裂只针对落入该区域的子集进行。

常用的预设停止条件包括：

- 当前节点中的样本输出值完全相同或几乎相同。
- 当前节点的样本无法通过特征进一步划分，即它们的特征值都相同。
- 当前节点的样本数量已经小于设定的最小阈值。
- 从根节点到当前节点的树深度已经达到预设的最大深度。
- 进一步划分所带来的增益小于指定阈值。

其中增益是指不再划分时的损失与进一步划分后的损失之差，其计算公式如下：

$$\Delta = \mathcal{L}(R) - \left(\frac{|R_1|}{|R|}\mathcal{L}(R_1) + \frac{|R_2|}{|R|}\mathcal{L}(R_2)\right) \tag{5.6}$$

其中 $|R|$ 表示区域 R 内的样本数量，而 R_1 和 R_2 则为将区域 R 按照最优划分方式得到的两个子区域。$\mathcal{L}(R)$ 表示落入区域 R 的样本集合对应的损失函数值，对于回归问题，可以使用公式5.2来进行计算。

在将增益作为停止条件时，需要注意其可能导致的“短视”问题。以图5.2所示的异或（XOR）为例，无论选择特征 x_1，还是 x_2 进行划分，增益始终为零（具体计算过程如图5.2的右侧所示）。这意味着，只要增益阈值设为任何大于零的数值，就不会执行任何划分。然而，这样的选择显然缺乏“远见”。事实上，如果先根据特征 x_1 进行第一次划分，再依据特征 x_2 进行进一步划分，就能够区分出四种不同的输入模式，从而构建出对应异或函数的决策树。

为了应对可能出现的“短视”现象，通常先放宽停止条件，从而将决策树构造成“完全生长”

的树，然后通过剪枝（Pruning）来防止对训练集的过拟合。剪枝过程一般从最靠近叶节点的中间节点开始，尝试合并分支（即剪去原先的分叉），并在验证集上评估合并是否有助于降低损失函数值。若合并能够降低验证集上的损失，则执行节点合并；否则，在该分支上停止剪枝。

在实际应用中，样本缺失部分特征是一种常见现象，并在决策树的训练和推理（使用）阶段都可能出现。下面分别讨论在训练和推理阶段如何处理特征缺失的问题。在训练集中，对于缺失特征的样本，常见的处理方法包括以下四种：

- 删除样本或特征。当缺失特征的样本比例较低（如低于 5%）时，可以直接删除这些样本。如果某一特征在大量样本中都缺失，则可将该特征整体移除，使其不参与模型构建。
- 对缺失特征进行填充。对于连续特征，常用均值或中位数进行填充。其中，中位数填充通常更加稳健，受异常值影响较小。对于离散特征，可以使用众数（出现频率最高的取值）进行填充。另一种常见做法是将缺失值视为一个独立类别，例如将缺省值设为“未知”。更为复杂的填充方法包括利用其他特征构建预测模型（如基于最近邻的方法）来推断缺失值。
- 构造替代特征。在确定当前节点的主分裂特征及其阈值时，只使用不含缺失特征的样本。随后，为该节点构建一组替代划分特征。主替代特征是指按照其切分方式对训练样本进行划分时，结果最接近主特征的切分结果。第二替代特征则是与主特征切分结果相似度排位第二的候选特征，依此类推。
- 基于概率分布的样本分配。对于在当前所选分裂特征上取值缺失的样本，可以根据该特征未缺失样本在各子节点中的分布比例，将其以相应的概率随机分配到不同子节点中。在推理阶段，当样本在预测路径上遇到缺失特征时，可以同时沿着所有可能的分支向下移动。样本到达叶节点之后，将获得来自多个叶节点的预测结果，最终的预测值通过对这些结果进行加权平均得到，其中权重由对应分支的样本分配概率确定。

在推理阶段，如果样本存在特征缺失，应当采用与训练阶段保持一致的处理策略，以避免训练与推理过程之间的不一致。具体而言，若在训练过程中构建了替代特征，则在主分裂特征缺失时，按照既定顺序依次使用这些替代特征进行划分。若训练阶段采用了基于概率分布的样本分配策略，则在推理过程中，缺失特征的样本会同时沿着所有可能的分支向下传播，并在到达叶节点之后，通过对各叶节点预测结果进行加权平均来得到最终预测值。若在训练阶段既未构建可用于近似划分的替代特征，也未采用基于概率分布的样本分配方法，则只能按照预先设定的方法对缺失特征进行填充后再进行推理。

5.3 分类树

对于分类问题，其基本思想与回归树类似，核心在于如何为逐步生长的决策树的每个节点选择合适的划分特征及其阈值。对于分类树，常用的分裂准则包括：

- **分类错误率**（Misclassification Error）：

$$\mathcal{L}(R_m) = \frac{1}{|R_m|} \sum_{\boldsymbol{x}_i \in R_m} \mathbb{I}(y_i \neq c_m) \tag{5.7}$$

其中 $|R_m|$ 表示区域 R_m 内的样本数量，y_i 表示样本 $\boldsymbol{x}_i$ 的真实类别标签，而 c_m 表示区域 R_m 中样本数最多的类别。该准则较为直观易懂，其目标是使得分裂后的分类错误率下降幅度最大。

- **信息熵**（Entropy）：

$$\mathcal{L}(R_m) = -\sum_{k=1}^{K} p_{mk} \log_2 p_{mk} \tag{5.8}$$

其中 K 表示分类任务中的类别总数，p_{mk} 表示区域 R_m 内样本属于类别 k 的概率，一般通过将该区域中属于类别 k 的样本数量除以 $|R_m|$ 来进行估计。我们在2.4节的讨论中已知，信息熵是衡量随机变量整体不确定性的指标。对于分类任务而言，如果落入区域 R_m 的样本在类别上的不确定性越高（即各类别的概率分布越均匀），信息熵的值就越大，此时使用该特征及其阈值对样本集合进行划分对目标分类的帮助越小；反之，如果区域内样本的类别不确定性越低（即类别分布越不均匀），信息熵的值就越小，则采用该特征进行划分对目标分类的贡献越大。

- **基尼指数**（Gini Index），又称基尼不纯度：

$$\mathcal{L}(R_m) = \sum_{k=1}^{K} p_{mk}(1 - p_{mk}) = 1 - \sum_{k=1}^{K} p_{mk}^2 \tag{5.9}$$

其中 p_{mk} 表示区域 R_m 内样本属于类别 k 的概率。直观上，基尼指数衡量了该区域内类别分布的“混杂程度”。当区域 R_m 完全纯（仅含单一类别）时，某个 $p_{mk} = 1$，基尼指数为零；当类别分布较为均匀时，p_{mk}^2 则较小，基尼指数较大，其最大值出现在各类别均匀分布的情况。最小化基尼指数相当于将样本划分到更加同质（纯）的节点，从而降低区域内样本的混杂程度。与信息熵相比，基尼指数无需进行对数计算（概率较小时对数运算可能导致数值不稳定），同时在处理大规模数据集时计算速度更快，这在遍历大量特征寻找最佳分裂点时尤其有利。值得注意的是，机器学习中的基尼指数与在经济学中主要用于衡量收入、财富等数值的分配不平等程度的基尼系数并不相同。

上述分类错误率、信息熵和基尼指数通常被称为节点不纯度（Impurity）度量指标，用于反映当前节点中样本的混杂程度。在二分类情况下，这三个指标随类别概率的变化如图5.3所示。从图中可以看出，信息熵和基尼指数的变化趋势较为相似，但相比于分类错误率，它们对类别概率的变化更加敏感，即概率的微小变化会导致信息熵或基尼指数发生较大变化。例如，考虑一个共有 800 个样本的二分类问题，每个类别各有 400 个样本。一种划分将样本分成 $(300, 100)$ 和 $(100, 300)$ 两个节点，而另一种划分将样本分成 $(200, 400)$ 和 $(200, 0)$ 两个节点。对于这两种划分，使用公式5.7计算，分类错误率均为 0.25：

$$\text{划分一：} \frac{400}{800} \times \frac{100}{400} + \frac{400}{800} \times \frac{100}{400} = 0.25$$

$$\text{划分二：} \frac{600}{800} \times \frac{200}{600} + \frac{200}{800} \times \frac{0}{200} = 0.25$$

显然，我们更倾向于选择划分二，因为其第二个节点 $(200, 0)$ 是完全纯的。如果使用信息熵作

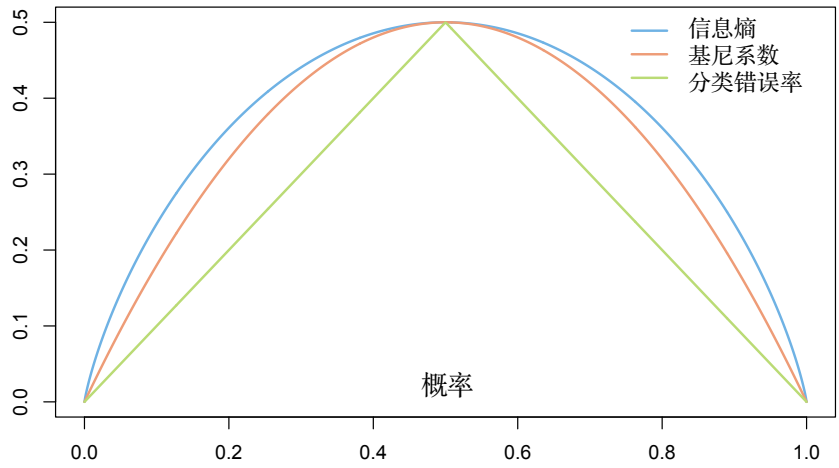


图 5.3: 二分类情况下节点不纯度度量对比：信息熵（经 0.5 倍缩放）、基尼指数和分类错误率。

为划分准则，划分一和划分二的损失分别为 0.8113 和 0.6887；而使用基尼指数时，损失分别为 0.3750 和 0.3333，这两种指标均优先选择划分二。因此，在决策树构造阶段，通常采用信息熵或基尼指数作为节点不纯度度量，以生成“完全生长”的树；而在剪枝阶段，则更多采用分类错误率（尽管这三个指标都可用作损失函数）来指导剪枝操作。

分类树的构建同样采用贪婪算法和二叉划分策略。首先从整个训练集开始，搜索所有可能将训练集划分为 R_1 和 R_2 两个部分的特征和分裂阈值，并且选择使下式最小的特征及其阈值来划分样本：

$$\frac{|R_1|}{|R|}\mathcal{L}(R_1)+\frac{|R_2|}{|R|}\mathcal{L}(R_2) \tag{5.10}$$

其中 $|R|$ 表示区域 R 中的样本数量，而 $\mathcal{L}(R)$ 表示所采用的损失函数（例如基尼指数或信息熵）。重复对所有节点进行划分，直到满足第5.2节介绍的预设停止条件。最后，利用验证集对树进行剪枝，以防止过拟合。完整的回归树与分类树构建过程如算法5.1所示。

表 5.1: 信用风险评估示例训练数据集

姓名	性别	年龄	收入	历史记录	信用风险
甲某	男	22	10,000	较差	高
乙某	男	50	8,000	中等	低
丙某	女	26	6,000	中等	高
丁某	男	45	50,000	良好	高
戊某	女	36	12,000	中等	低
己某	女	18	3,000	良好	高

我们以表5.1中给出的训练数据集为例，构建一个用于信用风险评估的决策树模型。该模型需要根据性别、年龄、收入和历史记录四个特征，对个体的信用风险等级进行预测。由于信用风险等级取离散值，因此该问题属于分类任务。在这四个特征中，性别和历史记录为离散特征。其中，历史记录的取值具有自然的顺序关系；性别虽然不具备顺序性，但训练数据中仅包含“男”

算法 5.1: 决策树构建算法

输入: 训练样本集、验证样本集、节点最少样本数 H、最大树深 T 和最小增益阈值 ϵ。

初始化: 创建一个树根节点;
while 仍有待处理节点 **do**
  任取一个待处理节点;
  if 当前节点内样本的输出值完全相同 **或者** 无可用于划分的特征 **then**
    将该节点标记为叶节点，并根据节点内样本计算其预测值。
  else
    if 节点样本数 $\geq H$ **并且** 当前节点的深度 $< T$ **then**
      搜索并选择使公式5.10取最小值的特征及其分裂阈值;
      if 分裂的增益 $\geq \epsilon$ **then**
        按选择的特征及其分裂阈值对当前节点进行划分;
        创建左子节点，并将该节点的深度记为其父节点深度加一;
        创建右子节点，并将该节点的深度记为其父节点深度加一;
      else
        将该节点标记为叶节点，并根据节点内样本计算其预测值。
    else
      将该节点标记为叶节点，并根据节点内样本计算其预测值。
使用验证集对树进行剪枝;

输出: 构建完成的决策树（包括各内部节点的划分特征、分裂阈值以及叶节点的预测值）。

和“女”两种取值，因此适合采用前述的二分划分策略。对于年龄和收入这些连续特征，我们以相邻样本特征值的平均值作为候选分裂阈值尝试划分。例如，当相邻样本的年龄取值分别为 18 和 22 时，可以将 $(18+22)/2=20$ 作为划分阈值。

假设采用信息熵作为节点分裂的准则，我们首先计算在不进行任何分裂、将所有样本视为一个整体时的初始信息熵:

$$\mathcal{L}(R) = -\frac{4}{6}\times\log_2\frac{4}{6} - \frac{2}{6}\times\log_2\frac{2}{6} = 0.9182$$

其中 $\frac{4}{6}$ 是数据集中信用风险为“高”的样本比例，而 $\frac{2}{6}$ 则是信用风险为“低”的样本比例。

接下来，我们考虑性别特征对目标分类的贡献。具体做法是：按照性别将训练样本划分为“男”性和“女”性两类，并分别计算每一类的信息熵。在“男”性类别中，目标分类的信息熵为:

$$\mathcal{L}(R(\text{性别}=\text{男})) = -\frac{2}{3}\times\log_2\frac{2}{3} - \frac{1}{3}\times\log_2\frac{1}{3} = 0.9182$$

其中 $\frac{2}{3}$ 是“男”性样本中信用风险为“高”的比例（即“甲某”和“丁某”），而 $\frac{1}{3}$ 则是“男”性样本中信用风险为“低”的比例（即“乙某”）。同样地，在“女”性类别中，目标分类的信息熵为:

$$\mathcal{L}(R(\text{性别}=\text{女})) = -\frac{2}{3}\times\log_2\frac{2}{3} - \frac{1}{3}\times\log_2\frac{1}{3} = 0.9182$$

然后，我们根据公式 5.10，按照“男”和“女”这两类样本在总样本中的比例，对划分后的两类

信息熵进行加权求和：

$$\frac{3}{6} \times 0.9182 + \frac{3}{6} \times 0.9182 = 0.9182$$

此时，我们可以根据公式5.6计算增益 Δ，即划分前后信息熵的变化量，具体计算如下：

$$\Delta(\text{性别}) = 0.9182 - 0.9182 = 0$$

由于增益 $\Delta(\text{性别}) = 0$，这表明在当前节点下，性别特征对目标分类（即信用风险）的预测没有任何贡献。在本例中，我们采用信息熵作为分裂准则，故此处的增益也称为信息增益，它量化了在引入某一特征信息后，目标分类不确定性（即信息熵）的下降程度。

对于年龄特征，最优的分裂阈值为 $(26 + 36)/2 = 31$ 岁。按照该阈值进行划分之后，两类的加权信息熵为 0.4591，相应的信息增益为：

$$\Delta(\text{年龄}) = 0.9182 - 0.4591 = 0.4591$$

对于收入特征，最优的分裂阈值为 $(6,000 + 8,000)/2 = 7,000$ 元。按该阈值进行划分之后，两类的加权信息熵为 0.6667，相应的信息增益为：

$$\Delta(\text{收入}) = 0.9182 - 0.6667 = 0.2515$$

对于历史记录特征，最优划分是将样本分为 {较差, 中等} 和 {良好} 两个子集。按照该划分，两类的加权信息熵为 0.6667，相应的信息增益为：

$$\Delta(\text{历史记录}) = 0.9182 - 0.6667 = 0.2515$$

综合以上计算结果，当前最优的划分特征是年龄，相应分裂阈值为 31 岁。按年龄特征进行划分后，左子节点（年龄 ≤ 31 岁）的样本集合为 {“甲某”, “丙某”, “己某”}，这些样本的信用风险均为 “高”，因此无需进一步划分；右子节点（年龄 > 31 岁）的样本集合为 {“乙某”, “丁某”, “戊某”}，其中部分样本的信用风险为 “高”，另一部分为 “低”，因此该节点仍需进一步划分。

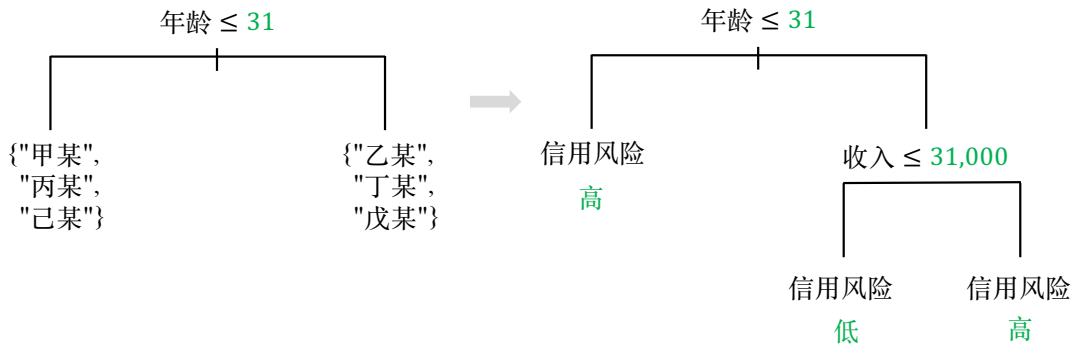
对于包含样本子集 {“乙某”, “丁某”, “戊某”} 的右子节点，我们需要重新计算在进一步分裂之前的信息熵：

$$\mathcal{L}(R(\text{年龄} > 31)) = -\frac{1}{3} \times \log_2 \frac{1}{3} - \frac{2}{3} \times \log_2 \frac{2}{3} = 0.9182$$

其中 $\frac{1}{3}$ 是该样本子集中信用风险为 “高” 的比例（即 “丁某”），而 $\frac{2}{3}$ 是信用风险为 “低” 的比例（即 “乙某” 和 “戊某”）。按照前述方法，我们对右子节点分别计算和评估使用性别、年龄、收入和历史记录特征进一步划分所带来的信息增益（注意，此处计算应基于该样本子集进行）。计算结果显示，使用收入特征（分裂阈值为 $(12,000 + 50,000)/2 = 31,000$）或历史记录特征（分成 {中等} 和 {良好} 两个子集）都能使得信息增益达到最大值 0.9182，即分裂之后的信息熵为零。最终，右子节点采用收入特征进行进一步划分的决策树如图5.4(b) 所示。

接下来，我们对决策树的优缺点进行总结。决策树主要具有以下优点：

- 具有较强的可解释性。从根节点到任一叶节点的路径都可以转化为一条明确的决策规则，从而直观地揭示输入特征取值与预测结果之间的关系。特别是当树深度较浅时，所总结的规则简洁明了。



(a) (b)

图 5.4: 采用表5.1的训练数据构建信用风险预测决策树的过程。(a) 使用年龄特征进行首次划分后的结果。(b) 最终构建完成的决策树。

- 同时处理离散型和连续型特征。既可以处理二值或多值的离散特征，也可以通过阈值划分处理连续特征，从而无需复杂的预处理即可直接应对现实中的混合类型数据。
- 对异常值具有一定鲁棒性。由于模型通过递归分裂将样本划分到不同子节点，异常值通常会被快速隔离在较小的分支或特定叶节点中。这种局部化的分裂机制可以限制异常值对模型整体结构和其它分支的影响。
- 能够应对特征缺失的样本。通过构建替代分裂特征、基于概率分布分配样本，或对离散特征增加 “未知” 类别等策略，决策树能够在不进行完整数据补全的情况下，充分利用包含缺失特征的样本信息。
- 对冗余或无关特征不敏感。在决策树的构建过程中，每一步分裂都会评估候选特征及其划分方式对预测目标的作用和贡献。对于与任务无关或贡献较小的特征，模型通常不会选择它们进行分裂，从而自然忽略无关特征的干扰。
- 特征值单调变换的不变性。决策树在确定分裂阈值时，主要依赖特征取值之间的相对顺序，而非具体的数值大小。因此，对输入特征施加单调变换（如线性缩放或对数变换）通常不会改变模型的结构和预测结果。
- 构建和训练效率较高。决策树采用自顶向下的贪婪算法与 “分而治之” 的策略进行构建。若假设分裂相对均衡，则随着树深度的增加，每次分裂处理的样本子集规模呈指数级缩小，从而整体计算复杂度较低。此外，虽然单棵树的生长过程具有序列性，但同一层级中不同节点的分裂相互独立，能够实现并行处理，可以进一步提高训练效率。相比于需要多次全局迭代收敛的模型（如支持向量机或神经网络），决策树通常只需有限次扫描数据集即可完成训练，在大规模数据集上训练有一定的速度优势。

“开箱即用”（Off-the-shelf）的方法指的是那些可以直接应用于各种数据情形和应用场景的方法，它们通常无需复杂耗时的数据预处理，也不需要对训练过程进行精细调校（如超参选择）。决策树非常接近理想的 “开箱即用” 方法，其具备良好的可解释性，能够同时处理离散型与连续型混合特征，对异常值较为鲁棒性，能够有效应对特征缺失，对无关特征不敏感，对特征值单调

变换保持不变性等特性。因此，决策树在实际应用中被广泛采用，并且常常作为基线（Baseline）方法，用于与其他方法进行性能比较。

然而，决策树也存在以下主要缺点：

- 预测性能可能受限。标准决策树在每次划分时通常仅考虑单个特征，虽然通过多层划分可以捕捉特征之间的交互关系，但对于复杂的特征组合模式，往往需要构建较深的树结构。这容易增加模型的复杂度和过拟合风险，从而影响单棵决策树的泛化性能。
- 结构的不稳定性。数据集中的微小扰动或噪声可能导致高层节点划分的变化。由于决策树是自顶向下逐层生成的，早期较上层的划分差异会向下累积并放大，最终可能产生显著不同的树结构。这种结构不稳定性也导致决策树预测结果的方差较高。
- 预测缺乏平滑性。在回归任务中，决策树通过分裂将特征空间划分为离散区域，并且对每个区域输出固定的预测值。这种分段式的预测会在不同区域之间产生阶梯式“跳跃”，难以有效拟合对连续性或平滑性要求较高的函数映射。

5.4 随机森林

如前所述，单棵决策树通常存在预测方差较高的缺点。为缓解这一问题，可以训练多棵决策树，并对它们的预测结果取平均，从而有效降低整体预测方差。假设每一棵决策树的预测方差均为 σ^2，若构建了 B 棵相互独立的决策树，并以其预测值的平均作为最终输出，则平均预测的方差可降为 $\frac{1}{B}\sigma^2$。

然而，如果都使用相同的训练数据集进行构建，所得到决策树的结构和预测也是相同的。若要构建 B 棵不同的决策树，首先需要获得 B 个彼此不同的训练数据集。一种直观的想法是将原数据集划分为 B 个不相交的子集，并且在每个子集上构建相应的决策树。但与完整数据集相比，每个子集所包含的样本数量将显著减少。这不仅会导致单棵决策树的预测方差明显增大（通常方差与样本量近似成反比），还会同时提高模型的预测偏差（表现为欠拟合）。此外，若数据中存在少数类或罕见模式，某些子集中可能完全缺失这些样本，使得对应的决策树无法学习到这些罕见模式。在最终通过平均得到的预测结果中，那些能够捕捉到罕见模式的树的预测很可能被多数未见过该模式的树所“淹没”，从而使罕见模式的正确预测难以在平均预测结果中胜出。

因此，通常会采用**自助法**（Bootstrap）来生成多个不同的训练数据集。具体做法是：从原始训练数据中进行有放回抽样，随机生成多个样本集，每个样本集的大小与原训练集相同。这一过程重复 B 次（例如 $B = 300$），从而得到 B 个不同的训练数据集。借助这些数据集，我们可以在每个样本集上分别训练一棵决策树，最终得到 B 棵互不相同的决策树。

上述通过对原始数据集进行有放回的重复抽样，从而生成多个模拟训练集的方法之所以被称为自助法（Bootstrap），其原因是 Bootstrap 一词的英文原义是：“To pull oneself up by one's bootstraps”，翻译成中文的字面含义是“拽着鞋带把自己拉起来”。这一行为在物理意义上显然是不可能的，但在语义上用来形容在不借助外部力量的情况下，仅凭自身条件完成看似不可能的任务。自助法的思想由美国斯坦福大学统计学家布拉德利 · 埃弗隆 (Bradley Efron) 于 1979 年引

入统计学领域。在传统的统计推断中，若希望估计某一统计量（如样本均值）的分布，通常需要从总体中获取更多相互独立的样本。然而在实际应用中，我们往往只能获得一个规模有限的原始数据集。为应对这一限制，自助法将已有样本视为对总体的近似，通过在该样本集上进行有放回的重复抽样，从而构造出多个“伪样本集”。自助法不依赖额外的外部信息，仅通过对现有数据的重采样，可以估计参数的分布特性及其不确定性。这种方法形象地体现了“拎着自己的鞋带把自己提了起来”的思想。

尽管每棵决策树是在不同的训练数据集上构建的，但它们之间仍存在一定的相关性，因此并非完全独立。对这些决策树的预测结果取平均后，其方差可以证明为：

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \tag{5.11}$$

其中 $\rho > 0$ 为任意两棵决策树之间的相关系数。设 B 棵决策树的预测结果分别为 $T_1, \ldots, T_B$，则它们的预测平均可以表示为：

$$\bar{T} = \frac{1}{B}\sum_{i=1}^{B} T_i \tag{5.12}$$

该预测平均值的方差为：

$$\begin{aligned}
\mathrm{Var}[\bar{T}] &= \mathrm{Var}\left[\frac{1}{B}\sum_{i=1}^{B} T_i\right] \\
&= \frac{1}{B^2}\mathrm{Var}\left[\sum_{i=1}^{B} T_i\right] && \text{（方差等于各个随机变量的方差之和} \\
&= \frac{1}{B^2}\left(\sum_{i=1}^{B}\mathrm{Var}[T_i] + \sum_{i\neq j}\mathrm{Cov}[T_i, T_j]\right) && \text{加上所有两两组合的协方差之和）} \\
&= \frac{1}{B^2}\left(B\sigma^2 + B(B-1)\rho\sigma^2\right) && \left(\rho = \frac{\mathrm{Cov}[T_i, T_j]}{\sqrt{\mathrm{Var}[T_i]}\sqrt{\mathrm{Var}[T_j]}}\right) \\
&= \frac{1}{B}\sigma^2 + \rho\sigma^2 - \frac{1}{B}\rho\sigma^2 \\
&= \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2
\end{aligned}$$

当树的数量 B 越来越多时，上式中的第二项会逐渐趋近于零，而第一项一直存在。因此，预测平均的方差主要受限于任意两棵不同决策树之间的相关系数。在单棵决策树方差不变的情况下，相关系数越小，预测平均的方差越小；反之，相关系数越大，预测平均的方差也越高。为了进一步降低不同决策树之间的相关性，在采用自助法生成不同训练数据集的基础上，每棵决策树只使用部分特征进行构建。具体而言，从所有 D 维特征中随机选择少于 D 个特征（通常分类任务约为 $\sqrt{D}$，而回归任务约为 $D/3$）来训练每棵树。每棵树所使用的特征数量是模型的超参数，可以根据具体任务进行调整。另一常用的策略是：在每个节点分裂时，重新随机抽取固定数量的候选特征，并仅在这些候选特征中搜索最佳分裂。不同节点可以抽到不同的特征子集。

值得注意的是，通过构建多棵决策树并将它们的预测结果取平均来作为最终预测的方法，

并不能降低预测的偏差。这是因为多棵树预测的平均值的期望等于单棵决策树预测的期望：

$$\mathbb{E}[\bar{T}] = \mathbb{E}\left[\frac{1}{B}\sum_{i=1}^{B} T_i\right] = \frac{1}{B}\sum_{i=1}^{B}\mathbb{E}[T_i] = \mathbb{E}[T_i]$$

因此，对具有相同期望的基预测器进行等权平均，本身不会改变其偏差。这里的主要收益来自方差的降低。

通过自助法生成多个训练数据集，进而在构建每棵决策树时随机选择固定数量的特征子集来降低不同树之间的相关性，最后以所有决策树预测结果的平均值作为最终预测的方法被称为**随机森林**（Random Forests）。对于分类任务，则将取平均操作替换为多数投票或加权投票方式。随机森林是**套袋法**（Bagging）或称**自助聚合法**（Bootstrap Aggregation）的一个典型实例，它以决策树作为集成框架下的基础模型。套袋法的核心思想是通过构建多个相对独立的模型，并对其预测结果进行集成（如取平均或多数投票）来抵消单个模型的随机误差，从而显著降低整体预测方差。它尤其适用于像决策树这样高方差、低偏差（易过拟合）的模型。

随机森林的一个重要特性是可以利用袋外样本（Out-of-bag Samples）来估计模型的预测误差，这类似于 K 折交叉验证，用于评估模型在未见样本上的性能。如图5.5所示，袋外样本是指在训练阶段未被用于构建某棵决策树的样本，即该样本不在该树的训练数据集中，因此它对于这棵决策树而言是未见样本。在不断构建新决策树的过程中，袋外样本误差可用于判断所需决策树的总数。当袋外样本误差趋于稳定时，训练过程可以终止，无需再构建更多的决策树。

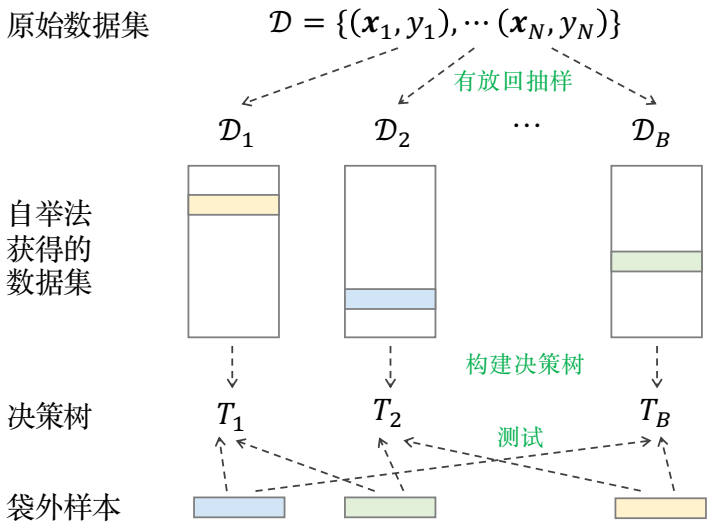


图 5.5: 随机森林中袋外样本误差估计。首先通过自助法对原始训练数据进行有放回抽样，生成多个不同的训练数据集，并分别训练多棵决策树。随后，对未被某棵决策树用于训练的样本（即袋外样本），在对应的决策树子集上进行预测，从而获得袋外样本误差的估计。

随机森林借助套袋法或自助聚合法克服了单棵决策树预测结果方差较高的缺点，并且显著提升了模型的预测性能。实践表明，随机森林是一种接近理想的“开箱即用”方法，能够在多种任务和应用场景中获得良好的效果，因此被广泛采用。

5.5 集成学习

集成学习（Ensemble Learning）的基本思想是将多个相对简单的基学习器（Base Learner）进行有效组合，充分发挥它们各自的优势与互补性，从而构建出预测性能更强的模型。集成学习主要涉及两个关键问题：一是如何利用训练数据构建一组基学习器；二是如何设计有效的组合策略，将这些基学习器的预测结果加以整合，形成最终的集成预测模型。

上一节介绍的随机森林是一种典型的集成学习方法，它以决策树作为基学习器，并通过套袋法（Bagging）将多棵决策树组合在一起，主要目的是通过集成来降低预测的方差。随机森林属于并行式集成，各基学习器之间相互独立，因此可以同时训练和构建。本节我们将着重讨论另一种串行式集成方法，即**提升法**（Boosting）。提升法逐轮加入新的基学习器，以降低所选损失函数。在平方损失回归中，这可以实现为拟合当前残差，从而通过逐步叠加新的基学习器来不断降低预测的偏差。更一般的梯度提升则拟合损失函数对当前预测值的负梯度。

提升法最初由美国普林斯顿大学计算机科学家罗伯特 · 沙皮尔 (Robert Schapire) 提出，用以回答“是否可以通过一组弱学习器构建出一个强学习器？”这一问题。他证明了对于某个复杂的分类任务，在满足弱可学习性和所需样本数量等条件时，如果能够找到比随机猜测略优的弱学习器（对于二分类问题，即准确率略高于 50% 的分类器），那么通过一定的构造性集成过程，就可以将这些弱学习器（Weak Learners）提升为具有任意高准确率的强学习器 [68]。

我们首先从一个相对简单的二分类问题入手，用以说明提升法如何通过构造性的集成过程，利用多个弱分类器逐步构建出一个强分类器。我们的目标是构造出一个函数或模型 $f(\boldsymbol{x})$，使其对输入 $\boldsymbol{x}$ 的预测结果 $\text{sign}(f(\boldsymbol{x}))$ 与真实标签 y 保持一致，其中 $y \in \{-1, +1\}$，而符号函数 $\text{sign}(z)$ 的定义为：

$$\text{sign}(z) = \begin{cases} +1 & \text{若 } z \geq 0 \\ -1 & \text{若 } z < 0 \end{cases} \tag{5.13}$$

在第3.6节中介绍用于二分类任务的 Logistic 回归模型时，我们已经指出，样本 $\boldsymbol{x}$ 属于类别 $\mathcal{C}_1$（即 $y = +1$）的后验概率可以表示为 $\sigma(z)$，其中 $\sigma(z)$ 为 Sigmoid 函数，而 z 为对数几率（Log Odds）。对数几率反映了样本 $\boldsymbol{x}$ 属于类别 $\mathcal{C}_1$ 与 $\mathcal{C}_2$（即 $y = -1$）的后验概率之比的对数，其定义为：

$$z = \ln\left(\frac{p(\mathcal{C}_1|\boldsymbol{x})}{p(\mathcal{C}_2|\boldsymbol{x})}\right) \tag{5.14}$$

此处，我们使用符号函数 $\text{sign}(z)$ 来代替 Sigmoid 函数，这同样是可行的。其原因在于，当样本 $\boldsymbol{x}$ 属于类别 $\mathcal{C}_1$ 的后验概率大于属于类别 $\mathcal{C}_2$ 的后验概率时，相应后验概率之比（即几率）大于 1，则对数几率为正；反之，则为负。结合使用符号函数，可以正确判断样本 $\boldsymbol{x}$ 的类别。由此可以看出，我们构造的函数 $f(\boldsymbol{x})$ 实际上就是用于估计对数几率。

我们需要选用合适的损失函数，以指导函数 $f(\boldsymbol{x})$ 的构造与优化过程。作为提升法中最具代表性的算法之一，**AdaBoost**（Adaptive Boosting）[26] 采用了如下的指数损失（Exponential Loss）

函数：

$$\mathcal{L}(y, f(\boldsymbol{x})) = \exp\left(-yf(\boldsymbol{x})\right) \tag{5.15}$$

指数损失函数之所以适用于二分类问题，其原因是在最小化期望指数损失的情况下，所得到的最优函数 $f(\boldsymbol{x})$ 恰好具有上述对数几率的形式。这意味着通过最小化指数损失，实际上等价于对类别后验概率的对数几率进行拟合。为了证明这一点，我们可以将指数损失的期望对函数 $f(\boldsymbol{x})$ 求偏导并令其为零，从而推导出这一结论。

$$\begin{aligned}
\frac{\partial \mathbb{E}\left[e^{-yf(\boldsymbol{x})}|\boldsymbol{x}\right]}{\partial f(\boldsymbol{x})} &= \frac{\partial\left[p(y=1|\boldsymbol{x})e^{-f(\boldsymbol{x})} + p(y=-1|\boldsymbol{x})e^{f(\boldsymbol{x})}\right]}{\partial f(\boldsymbol{x})} && \text{（期望的定义）} \\
&= -p(y=1|\boldsymbol{x})e^{-f(\boldsymbol{x})} + p(y=-1|\boldsymbol{x})e^{f(\boldsymbol{x})} \quad \Rightarrow \\
0 &= -p(y=1|\boldsymbol{x})e^{-f(\boldsymbol{x})} + p(y=-1|\boldsymbol{x})e^{f(\boldsymbol{x})} \quad \Rightarrow && \text{（偏导数置零）} \\
e^{2f(\boldsymbol{x})} &= \frac{p(y=1|\boldsymbol{x})}{p(y=-1|\boldsymbol{x})} \quad \Rightarrow \\
f(\boldsymbol{x}) &= \frac{1}{2}\ln\left(\frac{p(y=1|\boldsymbol{x})}{p(y=-1|\boldsymbol{x})}\right) = \frac{1}{2}\ln\left(\frac{p(\mathcal{C}_1|\boldsymbol{x})}{p(\mathcal{C}_2|\boldsymbol{x})}\right)
\end{aligned}$$

由此可见，严格来说，AdaBoost 算法实际上是让函数 $f(\boldsymbol{x})$ 拟合二分之一的对数几率。

在提升法中，函数 $f(\boldsymbol{x})$ 一般采用如下基函数 $\phi_j(\boldsymbol{x})$ 的加权和形式：

$$f(\boldsymbol{x}) = \sum_{j=1}^{M} \phi_j(\boldsymbol{x}) w_j \tag{5.16}$$

其中 w_j 是对应基函数的权重，而基函数 $\phi_j(\boldsymbol{x})$ 由弱学习器构成。典型的弱学习器包括：

- 决策树桩（Decision Stump）。深度仅为一的决策树，即只有一个分裂节点。也常采用稍深的决策树（深度在 3 到 8 层之间）作为基学习器。
- 浅层神经网络。通常为单层的人工神经网络，其输出可表示为 $\sigma(\boldsymbol{x}^\top \boldsymbol{w} + b)$，其中 $\sigma(z)$ 为非线性激活函数（例如 Sigmoid 函数），而 b 为可调偏置（Bias）。我们将在6章中详细介绍人工神经网络。
- 朴素贝叶斯（Naive Bayes）。假设在类别已知的条件下，各特征相互独立。因而样本 $\boldsymbol{x}$ 属于类别 $\mathcal{C}_k$ 的概率 $p(\mathcal{C}_k|\boldsymbol{x})$ 可以表示为 $p(\mathcal{C}_k)\prod_{i=1}^{D}(x_i|\mathcal{C}_k)$。参数通常通过最大似然估计求得。在提升法中，往往仅选取部分特征来构建朴素贝叶斯模型。

在之前已经介绍的线性模型中，模型通常先预设一组固定的基函数，然后通过优化算法确定每个基函数的权重。与线性模型不同，提升法采用逐步串行的策略来构造模型。具体而言，提升法首先构造第一个弱学习器 $f_1(\boldsymbol{x}) = \phi_1(\boldsymbol{x})$。对于二分类任务，通常可以取：

$$\phi_1(\boldsymbol{x}) = \frac{1}{2}\ln\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right) \tag{5.17}$$

其中 $\hat{\pi} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(y_i = 1)$。我们通常将数量较多的类别定义为 $y = 1$，以保证第一个弱分类器

的预测准确率高于 0.5。对于回归问题，通常取样本均值，即：

$$\phi_1(\boldsymbol{x}) = \bar{y} \tag{5.18}$$

经典的 AdaBoost 算法在初始阶段假设所有训练样本具有相同的重要性，即每个样本的重要性权重均为 $\frac{1}{N}$。随后，在当前样本权重分布下，通过优化指数损失函数训练首个弱分类器，使其在当前的训练集上实现尽可能高的分类准确率。具体训练过程将在下文中详述。

然后在迭代的第 $j \geq 2$ 步，我们计算下一个弱学习器的权重 w_j 及其参数 $\boldsymbol{\theta}_j$：

$$(w_j, \boldsymbol{\theta}_j) = \arg\min_{w,\boldsymbol{\theta}} \sum_{i=1}^{N} \mathcal{L}\left(y_i, f_{j-1}(\boldsymbol{x}_i) + w\phi(\boldsymbol{x}_i;\boldsymbol{\theta})\right) \tag{5.19}$$

同时更新模型为：

$$f_j(\boldsymbol{x}) = f_{j-1}(\boldsymbol{x}) + w_j\phi(\boldsymbol{x};\boldsymbol{\theta}_j) \tag{5.20}$$

此处，第 j 个弱分类器由其参数 $\boldsymbol{\theta}_j$ 来标识。值得注意的是，我们不会回头去调整之前已经加入模型的弱学习器的权重和参数，这显著加快了模型的构建速度。正因如此，这种方法被称为“前向逐步加法建模”（Forward Stagewise Additive Modeling）。该逐步迭代过程通常持续进行 M 步，其中 M 是这一方法的关键超参数。为了选择合适的 M 值，我们通常会在验证集上监控每一步迭代得到的模型性能。一旦在验证集上的性能开始下降，即停止模型构建过程，这一策略被称为“早停”（Early Stopping）。早停是对于预设的最大迭代步数而言的，可以有效防止模型过拟合。

我们将指数损失函数应用于上述迭代过程，则每步迭代会最小化以下损失：

$$\begin{aligned}
\mathcal{L}_j(\phi) &= \sum_{i=1}^{N} \exp\left(-y_i(f_{j-1}(\boldsymbol{x}_i) + w\phi(\boldsymbol{x}_i;\boldsymbol{\theta}))\right) \\
&= \sum_{i=1}^{N} \exp\left(-y_i f_{j-1}(\boldsymbol{x}_i)\right) \exp\left(-y_i w\phi(\boldsymbol{x}_i;\boldsymbol{\theta})\right) \\
&= \sum_{i=1}^{N} \alpha_i^{(j)} \exp\left(-y_i w\phi(\boldsymbol{x}_i;\boldsymbol{\theta})\right)
\end{aligned} \tag{5.21}$$

其中 $\alpha_i^{(j)} = \exp\left(-y_i f_{j-1}(\boldsymbol{x}_i)\right)$，这相当于在第 j 步迭代中，我们为样本 $\boldsymbol{x}_i$ 分配了一个重要性权重 $\alpha_i^{(j)}$。具体来说，当 y_i 与 $f_{j-1}(\boldsymbol{x}_i)$ 的符号相同时，说明 $f_{j-1}(\boldsymbol{x}_i)$ 对样本 $\boldsymbol{x}_i$ 的预测已经与其真实类别 y_i 一致，因此在当前迭代中该样本的重要性较低，对应的 $\alpha_i^{(j)}$ 值较小；而当 y_i 与 $f_{j-1}(\boldsymbol{x}_i)$ 的符号相反时，说明该样本当前被预测错误，其重要性权重 $\alpha_i^{(j)}$ 的值应较大，并且其大小与 $f_{j-1}(\boldsymbol{x}_i)$ 的绝对值成正比，表明在当前迭代中，应重点关注这些仍被错分的样本。

经典的 AdaBoost 或其离散版本（Discrete AdaBoost）要求每一步所加入的弱分类器输出为 $+1$ 或 -1，即 $\phi(\boldsymbol{x}) \in \{+1, -1\}$。在这种情况下，根据新加入的分类器 $\phi(\boldsymbol{x})$ 对样本的预测结果

正确与否，可以将公式5.21改写为：

$$\mathcal{L}_j(\phi) = e^{-w} \sum_{y_i=\phi(\boldsymbol{x}_i)} \alpha_i^{(j)} + e^{w} \sum_{y_i\neq\phi(\boldsymbol{x}_i)} \alpha_i^{(j)} \tag{5.22}$$

$$= \left(e^{w} - e^{-w}\right) \sum_{i=1}^{N} \alpha_i^{(j)} \mathbb{I}(y_i \neq \phi(\boldsymbol{x}_i)) + e^{-w} \sum_{i=1}^{N} \alpha_i^{(j)} \tag{5.23}$$

在以上公式5.23中，$(e^{w} - e^{-w})$ 项必定大于零（后文将给出证明），并且最后一项对于分类器 $\phi(\boldsymbol{x})$ 来说是常数。因此，使损失函数 $\mathcal{L}_j(\phi)$ 最小化的分类器为：

$$\phi_j(\boldsymbol{x}) = \arg\min_{\phi} \sum_{i=1}^{N} \alpha_i^{(j)} \mathbb{I}(y_i \neq \phi(\boldsymbol{x}_i)) \tag{5.24}$$

将该分类器代回公式5.22，并记：

$$A = \sum_{y_i=\phi(\boldsymbol{x}_i)} \alpha_i^{(j)}, \quad B = \sum_{y_i\neq\phi(\boldsymbol{x}_i)} \alpha_i^{(j)}$$

则公式5.22可简化为：

$$\mathcal{L}_j(\phi) = Ae^{-w} + Be^{w} \tag{5.25}$$

将上式关于权重 w 求导并置零，可得:

$$-Ae^{-w} + Be^{w} = 0 \quad \Rightarrow$$

$$Be^{w} = Ae^{-w} \quad \Rightarrow \tag{5.26}$$

$$e^{2w} = \frac{A}{B} \quad \Rightarrow$$

$$w = \frac{1}{2} \ln\left(\frac{A}{B}\right) \tag{5.27}$$

上式中的 $\frac{A}{B}$ 可以改写为：

$$\begin{aligned} \frac{A}{B} &= \frac{A}{A+B} \cdot \frac{A+B}{B} \\ &= \frac{(A+B)-B}{A+B} \cdot \frac{A+B}{B} \\ &= \left(1 - \frac{B}{A+B}\right) / \frac{B}{A+B} \end{aligned} \tag{5.28}$$

我们定义如下分类器 $\phi_j(\boldsymbol{x})$ 的加权错分率：

$$\text{Err}_j = \frac{\sum_{i=1}^{N} \alpha_i^{(j)} \mathbb{I}(y_i \neq \phi_j(\boldsymbol{x}_i))}{\sum_{i=1}^{N} \alpha_i^{(j)}} = \frac{B}{A+B} \tag{5.29}$$

结合公式5.29与5.28，公式5.27可以改写为:

$$w_j = \frac{1}{2} \ln\left(\frac{1 - \text{Err}_j}{\text{Err}_j}\right) \tag{5.30}$$

由于 Adaboost 要求所使用的弱分类器必须优于随机猜测，即满足 $\text{Err}_j < 0.5$（这意味着对数函数中分数的分母大于分子）。因此，公式5.30中的 w_j 必然大于零，进而每一步迭代中的 $(e^{w} - e^{-w})$ 也一定大于零。

算法 5.2: AdaBoost 算法（用于二分类任务，并采用指数损失函数）

输入: 训练样本集、验证样本集和最大迭代步数 M。

1 **初始化:** $\alpha_i = \frac{1}{N}, i = 1, \ldots, N$;

2 **for** $j \leftarrow 1$ **to** M **do**

3 利用训练集构建分类器: $\phi_j(\boldsymbol{x}) = \arg\min_{\phi} \sum_{i=1}^{N} \alpha_i \mathbb{I}(y_i \neq \phi(\boldsymbol{x}_i))$;

4 计算加权错分率: $\mathrm{Err}_j = \frac{\sum_{i=1}^{N} \alpha_i \mathbb{I}(y_i \neq \phi_j(\boldsymbol{x}_i))}{\sum_{i=1}^{N} \alpha_i}$;

5 计算新加入分类器 $\phi_j(\boldsymbol{x})$ 的权重: $w_j = \ln\left(\frac{1-\mathrm{Err}_j}{\mathrm{Err}_j}\right)$;

6 更新样本重要性权重: $\alpha_i \leftarrow \alpha_i \exp\left(w_j \mathbb{I}(y_i \neq \phi_j(\boldsymbol{x}_i))\right)$;

7 更新模型: $f_j(\boldsymbol{x}) = f_{j-1}(\boldsymbol{x}) + w_j \phi_j(\boldsymbol{x})$

8 **if** 当前模型在验证集上的性能出现下降 **then**

9 $M \leftarrow j$;

10 执行早停策略，提前结束迭代循环。

输出: $f(\boldsymbol{x}) = \mathrm{sign}\left(\sum_{j=1}^{M} \phi_j(\boldsymbol{x}) w_j\right)$。

随着新的弱分类器被引入，在下一次迭代中，训练样本的重要性权重将随之更新:

$$
\begin{aligned}
\alpha_i^{(j+1)} &= \exp\left(-y_i f_j(\boldsymbol{x}_i)\right) \\
&= \exp\left(-y_i \left(f_{j-1}(\boldsymbol{x}_i) + w_j \phi_j(\boldsymbol{x}_i)\right)\right) \quad \text{（根据公式5.20更新模型）} \\
&= \alpha_i^{(j)} e^{-w_j y_i \phi_j(\boldsymbol{x}_i)} \\
&= \alpha_i^{(j)} e^{w_j (2\mathbb{I}(y_i \neq \phi_j(\boldsymbol{x}_i)) - 1)} \quad \left(-y_i \phi_j(\boldsymbol{x}_i) = 2\mathbb{I}(y_i \neq \phi_j(\boldsymbol{x}_i)) - 1\right) \qquad (5.31) \\
&= \alpha_i^{(j)} e^{2 w_j \mathbb{I}(y_i \neq \phi_j(\boldsymbol{x}_i))} e^{-w_j} \qquad (5.32)
\end{aligned}
$$

在公式5.32中，因子 e^{-w_j} 对所有样本相同，只改变权重的整体尺度，因此在使用相对权重或归一化权重时可以省略。另外，公式5.31中的 $-y_i \phi_j(\boldsymbol{x}_i) = 2\mathbb{I}(y_i \neq \phi_j(\boldsymbol{x}_i)) - 1$ 成立的原因如下: 当 $y_i = \phi_j(\boldsymbol{x}_i)$ 时，有 $-y_i \phi_j(\boldsymbol{x}_i) = -1$；而当 $y_i \neq \phi_j(\boldsymbol{x}_i)$ 时，有 $-y_i \phi_j(\boldsymbol{x}_i) = +1$。因此，在这两种情形下，$-y_i \phi_j(\boldsymbol{x}_i)$ 与 $2\mathbb{I}(y_i \neq \phi_j(\boldsymbol{x}_i)) - 1$ 的值均一致。

至此，我们已经介绍了 AdaBoost 算法的所有主要步骤，其完整流程如算法5.2所示。在算法5.2中，由于变量 α_i 会在迭代中重复使用，因此无需再为其添加迭代步数的上标 (j) 以示区别。此外，算法第 5 行计算新加入分类器的权重时，相较于公式5.30少乘了一个 $\frac{1}{2}$。这是因为将所有分类器的权重整体缩小一半并不会影响最终组合分类器的结果。因此，算法第 6 行更新样本重要性权重时，指数项与公式5.32相差一个 2 倍的系数，这正是之前少乘的 $\frac{1}{2}$ 与指数中的 2 相互抵消的结果。需要注意的是，尽管提升法能够以串行方式集成诸如决策树桩等弱分类器，从而提升模型的整体预测性能，但这一过程也会失去决策树可解释性较好的这一重要优势。此外，由于在模型训练过程中对当前仍被错误预测的样本赋予更高的权重，提升法在一定程度上容易受到异常样本或噪声数据的影响。

在实际应用中，还会使用如下部分更新（Partial Updates）策略，通常可以获得更好的性能：

$$f_j(\boldsymbol{x}) = f_{j-1}(\boldsymbol{x}) + \nu w_j \phi(\boldsymbol{x}; \boldsymbol{\theta}_j) \tag{5.33}$$

其中 $0 < \nu < 1$ 是步长参数。在实际应用中，通常会选择一个较小的值（例如 $\nu = 0.1$）。这一策略称为“收缩”（Shrinkage）。

采用上述部分更新策略之所以能够提升模型的性能，是因为提升法可以被视为在函数空间中执行梯度下降，而参数 ν 则在其中扮演了学习率的角色。较小的学习率通常能够使优化过程更加平稳，避免参数更新过于激进而引发振荡，从而能够更精细地逼近损失函数的局部最小值。对于包含噪声或标注错误的数据集，当步长参数取 $\nu = 1$ 时，AdaBoost 算法往往会过度关注这些异常样本，进而导致过拟合。此外，选择较小的 ν 值可以使模型通过更多的迭代步骤逐步完成构造过程，这在效果上相当于引入了更多的弱学习器参与建模。随着弱学习器叠加数量的增加，集成效果更加充分。收缩减小每轮新增学习器的贡献，改变模型的拟合路径，并与迭代次数共同发挥正则化作用。

我们将上述方法推广到类别数 $K > 2$ 的多分类情形。在这种情况下，仅构建一个函数 $f(\boldsymbol{x})$ 是不够的，而需要为每个类别 $\mathcal{C}_k$ 分别构建相应的函数 $f_k(\boldsymbol{x})$，然后通过公式3.126所示的 Softmax 函数将这些 $f_k(\boldsymbol{x})$ 的输出转换为样本属于类别 $\mathcal{C}_k$ 的概率：

$$p(\mathcal{C}_k|\boldsymbol{x}) = \frac{e^{f_k(\boldsymbol{x})}}{\sum_{j=1}^{K} e^{f_j(\boldsymbol{x})}} \tag{5.34}$$

使用以上 Softmax 函数可以确保转换后的概率值落入 $[0, 1]$ 区间内，即 $0 \leq p(\mathcal{C}_k|\boldsymbol{x}) \leq 1$。然而，Softmax 函数有一个非常重要的平移不变性（Translation Invariance）。若对所有类别对应的得分函数 $f_k(\boldsymbol{x})$ 同时加上一个常数 h，则计算得到的概率 $p(\mathcal{C}_k|\boldsymbol{x})$ 保持不变：

$$p(\mathcal{C}_k|\boldsymbol{x}) = \frac{e^{f_k(\boldsymbol{x})+h}}{\sum_{j=1}^{K} e^{f_j(\boldsymbol{x})+h}} = \frac{e^h \cdot e^{f_k(\boldsymbol{x})}}{e^h \cdot \sum_{j=1}^{K} e^{f_j(\boldsymbol{x})}} = \frac{e^{f_k(\boldsymbol{x})}}{\sum_{j=1}^{K} e^{f_j(\boldsymbol{x})}} \tag{5.35}$$

上述平移不变性会导致模型的求解结果不唯一，即存在一个冗余的自由度。若不加以约束，$f_k(\boldsymbol{x})$ 的取值可能会整体不断增大，从而使指数项 $e^{f_k(\boldsymbol{x})}$ 快速增长，进而引发数值计算上的溢出问题，导致模型训练过程不稳定。为消除这一自由度，通常可以采用以下两种处理方式。第一种方式是选定其中一类作为参考类，例如约定 $f_K(\boldsymbol{x}) = 0$，其对应的指数函数值为 $e^{f_K(\boldsymbol{x})} = 1$。在这种设定下，其余各类别的函数得分 $f_k(\boldsymbol{x})$ 可以理解为相对于第 K 类的“差异得分”。因此，第 K 类的函数值起到了“锚点”的作用，其基本思想与经典二分类逻辑回归中的做法是类似的。第二种方式是直接对所有得分函数之和施加约束，例如要求 $\sum_{k=1}^{K} f_k(\boldsymbol{x}) = 0$。这种对称性约束避免了对某一特定类别的特殊处理，使得在统一的损失函数框架下可以同时优化所有类别的得分函数，从而在实现上更为简洁。之所以这种约束被称为“对称性约束”，是因为该约束方式在类别编号发生调换时仍保持不变，并未指定任何参考类或锚点类，对所有类别具有完全一致的约束形式。

对于多分类问题，常用的损失函数为交叉熵损失（Cross Entropy Loss）：

$$\mathcal{L}(y, p(\boldsymbol{x})) = -\sum_{k=1}^{K} \mathbb{I}(y = \mathcal{C}_k) \ln p(\mathcal{C}_k|\boldsymbol{x}) \tag{5.36}$$

交叉熵损失函数可以由以下似然函数推导得到。对于单个样本 $\boldsymbol{x}$，其似然函数为：

$$p(y|\boldsymbol{x}) = \prod_{k=1}^{K} p(\mathcal{C}_k|\boldsymbol{x})^{\mathbb{I}(y=\mathcal{C}_k)} \tag{5.37}$$

对以上似然函数取对数并加负号（即取负对数似然），即可得到交叉熵损失函数。对于似然函数，我们的优化目标是使其最大化，通过取负对数似然，优化目标等价地转化为最小化交叉熵损失。将 $p(\mathcal{C}_k|\boldsymbol{x})$ 的 Softmax 形式代入公式5.36，可得：

$$\mathcal{L}(y, f_k(\boldsymbol{x})) = -\sum_{k=1}^{K} \mathbb{I}(y = \mathcal{C}_k) f_k(\boldsymbol{x}) + \ln\left(\sum_{j=1}^{K} e^{f_j(\boldsymbol{x})}\right) \tag{5.38}$$

借助以上损失函数，我们可以直接对各个 $f_k(\boldsymbol{x})$ 函数进行优化。类别得分函数 $f_k(\boldsymbol{x})$ 同样采用公式 5.16 所示的加和形式，因此可以通过不断引入新的基函数，逐步迭代来优化其结果。此外，也可以不使用交叉熵损失，而是将 AdaBoost 算法中所使用的指数损失推广到多分类情形。具体方法可参见文献 [79]，此处不再展开讨论。

对于回归问题，常用的损失函数是平方误差损失：

$$\mathcal{L}(y, f(\boldsymbol{x})) = (y - f(\boldsymbol{x}))^2$$

在第 j 步迭代中，样本 $\boldsymbol{x}_i$ 的优化目标为：

$$\begin{aligned}\mathcal{L}(y_i, f_{j-1}(\boldsymbol{x}_i) + w\phi(\boldsymbol{x}_i;\boldsymbol{\theta})) &= (y_i - f_{j-1}(\boldsymbol{x}_i) - w\phi(\boldsymbol{x}_i;\boldsymbol{\theta}))^2 \\ &= (r_i^{(j)} - w\phi(\boldsymbol{x}_i;\boldsymbol{\theta}))^2\end{aligned} \tag{5.39}$$

其中 $r_i^{(j)} = y_i - f_{j-1}(\boldsymbol{x}_i)$ 表示当前模型在样本 $\boldsymbol{x}_i$ 上的残差（Residual）。因而，在平方误差损失下，新引入的函数项 $w\phi(\boldsymbol{x}_i;\boldsymbol{\theta})$ 用于对该残差进行拟合。通过不断引入新的基函数并对残差进行逼近，模型能够逐步降低整体预测误差。残差 $r_i^{(j)}$ 在回归问题中起到了类似于二分类问题中样本重要性权重 $\alpha_i^{(j)}$ 的作用。当某个样本的残差绝对值较大时，表明当前模型在该样本上的预测误差较大，因此在下一步迭代中会对该样本给予更多关注。这一点与分类问题中对仍被错分样本赋予更高权重、并使其在模型更新中发挥更大影响的机制是类似的。正是这种通过不断引入基学习器、逐步“蚕食”剩余误差的策略，构成了提升法（Boosting）成功的关键。通过迭代地修正前一阶段模型的不足，提升法能够有效降低模型整体的预测偏差，并将多个弱学习器逐步组合并提升成为性能更优的强学习器。

图5.6(a) 展示了三种常见的损失函数：分类错误率、指数损失和平方误差。首先，平方损失可以用于分类，但它会惩罚类别方向正确而数值幅度过大的输出。即使预测结果已经正确，例如真实标签为 $+1$，如果模型的预测值偏离真实值过远（如预测值为 10），平方误差仍会产生较大损失，迫使模型将预测值拉回至 1 附近，从而引入不必要的误差修正。分类任务主要关注预测类别是否正确，而平方误差关注预测值与真实值的数值差距，两者在目标上并不完全一致。分类错误率在所有训练样本都被正确分类时，将不再变化，而指数损失仍可能继续下降。具体来说，当 $yf(\boldsymbol{x}) < 0$ 时，样本 $\boldsymbol{x}$ 仍被分类错误；而当 $yf(\boldsymbol{x}) > 0$ 时，样本 $\boldsymbol{x}$ 已被正确分类。决策边界位于 $f(\boldsymbol{x}) = 0$ 处。经归一化后的 $yf(\boldsymbol{x})$ 可以看作样本的分类间隔（Margin）。指数损失的单

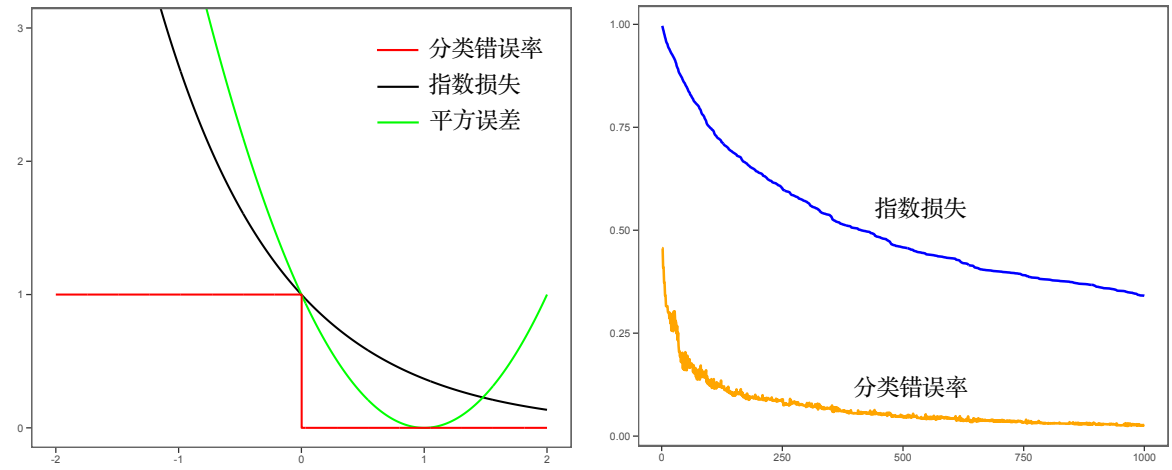


图 5.6: 损失函数比较。(a) 二分类问题中三种损失函数。真实标签为 $y \in \{-1, +1\}$，预测结果为 $\text{sign}(f(\boldsymbol{x}))$，其中 $f(\boldsymbol{x})$ 为预测模型。横轴表示 $yf(\boldsymbol{x})$，纵轴表示损失函数值。损失函数包括：分类错误率 $\mathbb{I}(y \neq \text{sign}(f(\boldsymbol{x})))$，指数损失 $\exp(-yf(\boldsymbol{x}))$ 以及平方误差 $(y - f(\boldsymbol{x}))^2$。(b)AdaBoost 算法训练过程中损失随迭代步数的变化（以决策树桩为弱学习器）。结果显示，当分类错误率在约 800 步迭代时趋近于零，但指数损失仍然持续下降。

调递减的性质允许通过增加 $yf(\boldsymbol{x})$ 的值来进一步降低损失，这相当于扩大类别之间的间隔。较大的分类间隔通常意味着样本的分类结果具有更高的稳定性。直观地说，分类间隔可以为样本提供一定的 “缓冲区”：在其他条件相近的情况下，即使未见样本相较于训练样本存在一定程度的偏移，具有较大间隔的模型也更有可能保持正确分类；而间隔较小的样本则更容易因输入变化而跨越决策边界，从而导致错分。因此，从图5.6(b) 中可以观察到（基于由 10 个输入特征构成、每类 1,000 个样本的二分类模拟数据），尽管分类错误率已经接近零，指数损失仍然在持续下降。这表明模型仍可通过进一步扩大类间间隔（类似于支持向量机的原理）来提升其鲁棒性与泛化性能。这也解释了为何在 AdaBoost 中，即便训练样本上的分类错误率已归零，测试误差有时仍有可能继续下降。

我们已经介绍了并行构建的套袋法（Bagging）和串行构造的提升法（Boosting）。另一种重要的集成学习方法是**堆叠法**（Stacking），其主要特征是在模型中引入分层结构。堆叠法首先训练多个结构或性质不同的基模型（如支持向量机、K 近邻、随机森林等），随后将这些基模型的输出作为 “特征”，输入到更高一层的元模型（Meta-model）中完成最终预测。为了缓解过拟合，元学习器通常采用结构相对简单的模型，例如逻辑回归或线性回归。

为了避免信息泄漏，元学习器不能直接使用基模型在训练集上的预测结果作为输入。构建元学习器的关键在于利用折外数据（Out-of-fold）预测。其标准流程如下：首先将训练数据划分为 K 个互不重叠的子集（类似于 K 折交叉验证）；随后在每一次迭代中，使用其中 $K-1$ 份数据训练基模型，并对剩余的 1 份数据进行预测。重复该过程 K 次，确保训练集中的每个样本都有一个未使用其训练的模型所生成的预测值；接着，将这些折外数据的预测值作为输入特征，

并以其真实值作为预测目标，用于训练元学习器；最后，在元模型训练完成后，利用全部训练数据重新训练各基模型。在预测阶段，待测数据首先经过各基模型生成预测值，再输入已训练好的元学习器，得出最终结论。堆叠法的主要优势在于能够整合不同类型学习算法的互补优势，从而捕捉数据中更复杂的模式。

主要符号表

数学符号

$\boldsymbol{x}$	向量
y	标量
D	向量维度
x_j	向量的第 j 个分量（向量分量为标量）
f	输入到输出的某种映射
$\mathbf{X}^\top$	矩阵转置
$\mathbf{X}^{-1}$	逆矩阵
$\|\mathbf{X}\|$	矩阵的行列式
$\mathrm{tr}(\mathbf{X})$	矩阵的迹（矩阵所有特征值之和，也等于主对角线元素的总和）
$\ln$	以自然常数 e 为底的对数
e^x 或 $\exp(x)$	自然常数 e 的指数函数
$f'(x)$	一阶导数
$f''(x)$	二阶导数
$\nabla f(\boldsymbol{x})$	一阶梯度（梯度向量）
$\nabla^2 f(\boldsymbol{x})$	二阶梯度（海森矩阵）
$\frac{\partial f(x)}{\partial x}$	一阶偏导
$\frac{\partial^2 f(x)}{\partial x^2}$	二阶偏导

机器学习

$\mathcal{D}$	数据集
N	样本数量
$\boldsymbol{x}_i$	第 i 个训练样本
y_i	第 i 个训练样本对应的输出
$\kappa(\boldsymbol{x}, \boldsymbol{x}')$	核函数
$\mathcal{L}(\cdot)$	损失函数
$\mathcal{J}(\cdot)$	目标函数或代价函数

概率统计

$\mathbb{E}[x]$	均值或期望
$\text{Var}[x]$	方差
$\text{Cov}[x, z]$	随机变量 x 和 z 之间的协方差
$\Gamma(z)$	伽玛函数
$\mathcal{N}(x\|\mu, \sigma^2)$	高斯分布，其中 μ 为均值，σ 为标准差
$\mathcal{N}(x\|\mu, \lambda^{-1})$	高斯分布，其中 μ 为均值，λ 为精度
$\text{Bern}(x\|\mu)$	伯努利分布
$\text{Bin}(k\|\mu, N)$	二项分布
$\text{Beta}(\mu\|\alpha, \beta)$	贝塔分布
$\text{Poisson}(k\|\lambda)$	泊松分布
$\text{Exp}(\lambda\|\beta)$	指数分布
$\text{Gamma}(\lambda\|\alpha, \beta)$	伽玛分布，其中 α 为形状参数，β 为逆尺度参数
$\text{Mult}(\boldsymbol{m}\|\boldsymbol{\mu}, N)$	多项分布
$\text{Dir}(\boldsymbol{\mu}\|\boldsymbol{\alpha})$	狄利克雷分布
$\mathcal{N}(\boldsymbol{x}\|\boldsymbol{\mu}, \boldsymbol{\Sigma})$	多维高斯分布，其中 $\boldsymbol{\mu}$ 为均值向量，$\boldsymbol{\Sigma}$ 为协方差矩阵
$\mathcal{N}(\boldsymbol{x}\|\boldsymbol{\mu}, \boldsymbol{\Lambda})$	多维高斯分布，其中 $\boldsymbol{\mu}$ 为均值向量，$\boldsymbol{\Lambda}$ 为精度矩阵
$\mathcal{W}(\boldsymbol{\Lambda}\|\mathbf{W}, \eta)$	威沙特分布，其中 $\mathbf{W}$ 为尺度矩阵，η 为自由度
$\text{NormalGamma}(\mu, \lambda\|\upsilon, \nu, \alpha, \beta)$	高斯-伽玛分布
$\text{NormalWishart}(\boldsymbol{\mu}, \boldsymbol{\Lambda}\|\boldsymbol{\upsilon}, \nu, \mathbf{W}, \eta)$	高斯-威沙特分布
$\text{St}(x\|\mu, \tau, \eta)$	学生氏分布，其中 μ 为均值，τ 为精度，η 为自由度
$\text{St}(\boldsymbol{x}\|\boldsymbol{\mu}, \boldsymbol{\Lambda}, \eta)$	多维学生氏分布，其中 $\boldsymbol{\mu}$ 为均值向量，$\boldsymbol{\Lambda}$ 为精度矩阵
$\text{Laplace}(x\|\mu, \tau)$	拉普拉斯分布，其中 μ 为位置参数，τ 为尺度参数
$\chi^2(x\|\eta)$	卡方分布，其中 η 为自由度
$\text{U}(x\|a, b)$	连续型均匀分布，其中 a 为下界、b 为上界 $(a < b)$
$\text{H}[x]$	信息熵，其中 x 为随机变量
$\text{H}[y\|x]$	条件熵，其中 x 和 y 为随机变量
$\text{H}[p, q]$	分布 p 和 q 之间的交叉熵
$D_{\text{KL}}(p\|\|q)$	分布 p 和 q 之间的 KL 散度

参考文献

[1] Daniel J. Amit, Hanoch Gutfreund, and Haim Sompolinsky. “Storing infinite numbers of patterns in a spin-glass model of neural networks”. In: *Physical Review Letters* 55.14 (1985), pp. 1530–1533.

[2] Mikel Artetxe et al. “On the role of bidirectionality in language model pre-training”. In: *Findings of the Conference on Empirical Methods in Natural Language Processing* (2022), pp. 3973–3985.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. “Layer normalization”. In: *arXiv: 1607.06450* (2016).

[4] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166.

[5] Jame O. Berger. *Statistical decision theory and Bayesian analysis (second edition)*. Springer, 1980.

[6] James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization”. In: *Journal of Machine Learning Research* 13.2 (2012).

[7] Jeremy Bernstein and Laker Newhouse. “Old optimizer, new norm: An anthology”. In: *Proceedings of the 16th Annual Workshop on Optimization for Machine Learning* (2024), pp. 1–19.

[8] J Besag. “On spatio-temporal models and Markov fields”. In: *Transactions of the 7th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes* (1974), pp. 47–75.

[9] Julian Besag. “On the statistical analysis of dirty pictures”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 48.3 (1986), pp. 259–279.

[10] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[11] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. “A training algorithm for optimal margin classifiers”. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. 1992, pp. 144–152.

[12] George E.P. Box and Norman Richard Draper. *Empirical model-building and response surfaces*. John Wiley & Sons, 1987.

[13] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[14] Tom Brown et al. “Language models are few-shot learners”. In: *Proceedings of the Conference on Neural Information Processing Systems* 33 (2020), pp. 1877–1901.

[15] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder–decoder for statistical machine translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (2014), pp. 1724–1734.

[16] Peter Clifford. “Markov random fields in statistics”. In: *Disorder in Physical Systems: A Volume in Honour of John M. Hammersley* (1990), pp. 19–32.

[17] Thomas H. Cormen et al. *Introduction to algorithms (third edition)*. The MIT Press, 2022.

[18] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley, 1991.

[19] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[20] Morris H. DeGroot and Mark J. Schevish. *Probability and statistics (forth edition)*. China Machine Press, 2012.

[21] Jacob Devlin et al. “BERT: Pre-training of deep bidirectional transformers for language understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2019), pp. 4171–4186.

[22] Alexey Dosovitskiy et al. “An image is worth 16×16 words: Transformers for image recognition at scale”. In: *Proceedings of the International Conference on Learning Representations* (2021).

[23] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive subgradient methods for online learning and stochastic optimization”. In: *Journal of Machine Learning Research* 12.7 (2011).

[24] Walter D. Fisher. “On grouping for maximum homogeneity”. In: *Journal of the American Statistical Association* 53.284 (1958), pp. 789–798.

[25] Roger Fletcher. *Practical methods of optimization (second edition)*. Wiley, 1987.

[26] Yoav Freund and Robert E. Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting”. In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139.

[27] Stuart Geman and Donald Geman. “Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images”. In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1984), pp. 721–741.

[28] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. “Learning precise timing with LSTM recurrent networks”. In: *Journal of Machine Learning Research* 3 (2002), pp. 115–143.

[29] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (2010), pp. 249–256.

[30] Ian J Goodfellow et al. “Generative adversarial nets”. In: *Proceedings of the Conference on Neural Information Processing Systems* 27 (2014).

[31] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction (second edition)*. Springer, 2009.

[32] Kaiming He and Jian Sun. “Convolutional neural networks at constrained time cost”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 5353–5360.

[33] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

[34] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1026–1034.

[35] Nicholas J. Higham. *Functions of matrices: Theory and computation*. SIAM, 2008.

[36] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv: 1503.02531* (2015).

[37] Geoffrey E. Hinton. “Training products of experts by minimizing contrastive divergence”. In: *Neural Computation* 14.8 (2002), pp. 1771–1800.

[38] John J. Hopfield. “Neural networks and physical systems with emergent collective computational abilities”. In: *Proceedings of the National Academy of Sciences* 79.8 (1982), pp. 2554–2558.

[39] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *Proceedings of the International Conference on Machine Learning* (2015), pp. 448–456.

[40] Arieh Iserles. *A first course in the numerical analysis of differential equations*. 44. Cambridge university press, 2009.

[41] Max Jaderberg et al. “Population based training of neural networks”. In: *arXiv: 1711.09846* (2017).

[42] Kevin Jamieson and Ameet Talwalkar. “Non-stochastic best arm identification and hyperparameter optimization”. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (2016), pp. 240–248.

[43] Keller Jordan et al. *Muon: An optimizer for hidden layers in neural networks*. 2024. URL: https://kellerjordan.github.io/posts/muon/.

[44] Diederik P. Kingma and Jimmy L. Ba. “Adam: A method for stochastic optimization”. In: *Proceedings of the International Conference on Learning Representations* (2015).

[45] Josef Kittler and Janos Föglein. “Contextual classification of multispectral pixel data”. In: *Image and vision computing* 2.1 (1984), pp. 13–29.

[46] Solomon Kullback and Richard A. Leibler. “On information and sufficiency”. In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86.

[47] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (2002), pp. 2278–2324.

[48] Mike Lewis et al. “BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), pp. 7871–7880.

[49] Yaron Lipman et al. “Flow matching guide and code”. In: *arXiv: 2412.06264* (2024).

[50] Ilya Loshchilov and Frank Hutter. “Decoupled weight decay regularization”. In: *arXiv: 1711.05101* (2017).

[51] Calvin Luo. “Understanding diffusion models: A unified perspective”. In: *arXiv: 2208.11970* (2022).

[52] Marvin Minsky and Seymour Papert. *Perceptrons: An introduction to computational geometry*. The MIT Press, 1969.

[53] Kevin P. Murphy. *Machine learning: A probabilistic perspective*. The MIT Press, 2012.

[54] Brady Neal et al. “A modern take on the bias-variance tradeoff in neural networks”. In: *arXiv: 1810.08591* (2018).

[55] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Proceedings of the Conference on Neural Information Processing Systems* 35 (2022), pp. 27730–27744.

[56] Judea Pearl. *Probabilistic reasoning in intelligent systems*. Morgan kaufmann, 2014.

[57] Matthew E. Peters et al. “Deep Contextualized Word Representations”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2018), pp. 2227–2237.

[58] John C. Platt. “Fast training of support vector machines using sequential minimal optimization”. In: *Advances in Kernel Methods* (1999), pp. 185–208.

[59] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *Proceedings of the International Conference on Learning Representations* (2016).

[60] Alec Radford et al. “Improving language understanding by generative pre-training”. In: *OpenAI’s Technical Report* (2018).

[61] Colin Raffel et al. “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67.

[62] Scott Reed et al. “Generative adversarial text to image synthesis”. In: *Proceedings of the International Conference on Machine Learning* (2016), pp. 1060–1069.

[63] Edmund T. Rolls. *Cerebral cortex: principles of operation*. Oxford University Press, 2016.

[64] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional networks for biomedical image segmentation”. In: *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention* (2015), pp. 234–241.

[65] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323.6088 (1986), pp. 533–536.

[66] Leonard J. Savage. *The subjective basis of statistical practice*. University of Michigan, 1961.

[67] Andrew M Saxe, James L McClelland, and Surya Ganguli. “Exact solutions to the nonlinear dynamics of learning in deep linear neural networks”. In: *Proceedings of the International Conference on Learning Representations* (2014).

[68] Robert E. Schapire. “The strength of weak learnability”. In: *Machine learning* 5.2 (1990), pp. 197–227.

[69] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural machine translation of rare words with subword units”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (2016), pp. 1715–1725.

[70] Claude Elwood Shannon. “A mathematical theory of communication”. In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423.

[71] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. “Practical Bayesian optimization of machine learning algorithms”. In: *Proceedings of the Conference on Neural Information Processing Systems* 25 (2012).

[72] Yang Song et al. “Score-based generative modeling through stochastic differential equations”. In: *Proceedings of the International Conference on Learning Representations* (2021).

[73] Gilbert Strang. *Introduction to linear algebra*. Wellesley-Cambridge Press, 2016.

[74] Jianlin Su et al. “Roformer: Enhanced transformer with rotary position embedding”. In: *Neurocomputing* 568 (2024), p. 127063.

[75] Tijmen Tieleman and Geoffrey E. Hinton. “Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural Networks for Machine Learning* 4.2 (2012), pp. 26–26.

[76] Antonio Torralba, Phillip Isola, and William Freeman. *Foundations of computer vision*. The MIT Press, 2024.

[77] Ashish Vaswani et al. “Attention is all you need”. In: *Proceedings of the Conference on Neural Information Processing Systems* 30 (2017).

[78] Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. “Deep learning for Chinese word segmentation and POS tagging”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (2013), pp. 647–657.

[79] Ji Zhu et al. “Multi-class AdaBoost”. In: *Statistics and its Interface* 2.3 (2009), pp. 349–360.

[80] Hui Zou and Trevor Hastie. “Regularization and variable selection via the elastic net”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67.2 (2005), pp. 301–320.