Author：Liu Jian

Time：2020-06-07

机器学习8a-集成学习

1 集成学习的思路

集成学习 (ensemble learning) 也称多分类器系统 (multi-classifier system)、基于委员会的学习 (committee-based learning)，它通过构建并结合多个个体学习器 (individual learner) 来完成学习任务。若集成的个体学习器类型相同，则集成是同质的 (homogeneous)，此时也称个体学习器为基学习器 (base learner)，基学习器的学习算法也被称为基学习算法 (base learning algorithm)；若集成的个体学习器类型不同，则集成是异质的 (heterogenous)，此时也称个体学习器为组件学习器 (component learner)。

集成学习的理论基础是在 PAC 可学习框架下弱学习器 (wweak learner) 与强学习器 (strong learner) 等价。弱学习器，也就是泛化性能略优于随机猜测的学习器能通过组合的方式变成强学习器，通俗地理解就是三个臭皮匠顶个诸葛亮。但是，要获得好的集成，个体学习器应“好而不同”：“好”是指个体学习器要有一定的准确性，应至少不差于弱学习器；“不同”是指个体学习器应存在差异，具有多样性 (diversity)。一般地，准确性很高之后，要增加多样性就需要牺牲准确性，如何产生并结合“好而不同”的个体学习器是集成学习研究的核心，即**集成学习的关键问题为：(1) “好而不同”个体学习器的生成；(2) 个体学习器的集成。一般地，“好”的个体学习器由其学习算法是很容易保证的，关键在于如何实现个体学习器的“不同”，也就是多样性。接下来，本章将讨论多样性及集成的一般策略，具体的集成学习算法在下面两章介绍。**

1.1 多样性

1.1.1 回归任务的误差-分歧分解 (error-ambiguity decomposition)：

本小节通过误差-分歧分解来解释为什么个体学习器准确性越高，多样性越大，则集成越好。注意下面推导只适用于回归学习，难以直接推广到分类学习任务上去。

我们要学习的目标概念 $c_{obj}:\mathbb{R}^d\to\mathbb{R}$， d 为输入属性种数，个体学习器 $h_i(\cdot),\ i=1,\cdots,T$，个体学习器的集成 $H(\boldsymbol{x})=\sum_{i=1}^T w_i h_i(\boldsymbol{x}),\ w_i\geqslant 0,\sum_{i=1}^T w_i=1$，概率密度 $p(\boldsymbol{x})$。

- 第 i 个学习器 $h_i(\cdot)$ 在某个输入 (也就是示例) $\boldsymbol{x}$ 上的分歧定义为：

$$A_i(\boldsymbol{x}) = (h_i(\boldsymbol{x}) - H(\boldsymbol{x}))^2$$

在该输入 $\boldsymbol{x}$ 上分歧的集成为：

$$A_{1\cdots T}(\boldsymbol{x}) = \sum_{i=1}^{T} w_i A_i = \sum_{i=1}^{T} w_i (h_i(\boldsymbol{x}) - H(\boldsymbol{x}))^2$$

遍历整个输入空间，可得总分歧为：

$$A = \int A_{1\cdots T}(\boldsymbol{x}) p(\boldsymbol{x}) \mathrm{d}\mathbf{x}$$

可以看到，**总分歧 A 表征了个体学习器在整个输入空间上的不一致性，即多样性。**

- 损失函数为均方损失，则第 i 个学习器在某个输入 $\boldsymbol{x}$ 上的损失为：

$$L_i(\boldsymbol{x}) = (h_i(\boldsymbol{x}) - c_{obj}(\boldsymbol{x}))^2$$

集成学习器 H 在 $\boldsymbol{x}$ 上的损失为：

$$L_H(\boldsymbol{x}) = (H(\boldsymbol{x}) - c_{obj}(\boldsymbol{x}))^2$$

第 i 个学习器 h_i 和集成学习器 H 的泛化误差分别为：

$$R(h_i) = \int L_i(\boldsymbol{x}) p(\boldsymbol{x}) \mathrm{d}\mathbf{x}$$

$$R(H) = \int L_H(\boldsymbol{x}) p(\boldsymbol{x}) \mathrm{d}\mathbf{x}$$

- 易证：

$$A_{1\cdots T}(\boldsymbol{x}) = \sum_{i=1}^{T} w_i (h_i(\boldsymbol{x}) - H(\boldsymbol{x}))^2$$

$$= \left(\sum_{i=1}^{T} w_i (h_i(\boldsymbol{x}) - c_{obj}(\boldsymbol{x}))^2 \right) - (H(\boldsymbol{x}) - c_{obj}(\boldsymbol{x}))^2$$

$$= \left(\sum_{i=1}^{T} w_i L_i(\boldsymbol{x}) \right) - L_H(\boldsymbol{x})$$

则：

$$A = \int A_{1\cdots T}(\boldsymbol{x}) p(\boldsymbol{x}) \mathrm{d}\mathbf{x}$$

$$= \int \left(\left(\sum_{i=1}^{T} w_i L_i(\boldsymbol{x}) p(\boldsymbol{x}) \right) - L_H(\boldsymbol{x}) \right) p(\boldsymbol{x}) \mathrm{d}\mathbf{x}$$

$$= \sum_{i=1}^{T} w_i R(h_i) - R(H)$$

即**集成学习器的泛化误差 $R(H)$ 等于各个体学习器的泛化误差的加权平均 $\sum_{i=1}^{T} w_i R(h_i)$ 减去总分歧 A**：

$$R(H) = \left(\sum_{i=1}^{T} w_i R(h_i) \right) - A$$

这也就证明了结论：个体学习器准确性越高，多样性越大，则集成越好。那么，**我们能不能直接以 $\left(\sum_{i=1}^{T} w_i R(h_i)\right) - A$ 为优化目标进行集成学习呢？但遗憾的是，现实任务中很难直接对其进行优化，不仅因为它们定义在整个样本空间上 (对于这个问题，可采用经验估计量作为近似，类似于期望风险最小化和经验风险最小化的关系)，还因为 A 不是一个可直接操作的多样性度量，它仅在集成构造好之后才能进行估计。**

1.1.2 多样性度量 (diversity measure) 及增强

本小节首先讨论衡量多样性的指标，然后讨论增强多样性的策略。

- **多样性度量**也称差异性度量，典型做法是考察个体分类器两两相似/不相似性。给定样本集 $S = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)\}$，对于二分类任务 $y_i \in \{-1, +1\}$，分类器 h_i, h_j 的预测结果列量表 (contingency table) 为：

--	$h_i = +1$	$h_i = -1$
$h_j = +1$	a	c
$h_j = -1$	b	d
其中，a, b, c, d 为对应情况的样本个数，$a + b + c + d = N$。常见的多样性度量有：		

(1) 不合度量 (disagreement measure)：

$$\mathrm{dis}_{ij} = \frac{b + c}{N} \in [0, 1]$$

(2) 相关系数 (correlation coefficient)：

$$\rho_{ij} = \frac{ad - bc}{\sqrt{(a + b)(a + c)(c + d)(b + d)}} \in [-1, 1]$$

(3) Q 统计量 (Q-statistic)：

$$Q_{ij} = \frac{ad - bc}{ad + bc}$$

可以看到，Q_{ij} 与 ρ_{ij} 符号相同，且 $|Q_{ij}| \geqslant |\rho_{ij}|$。

(4) κ -统计量 (κ -statistic)：

$$\kappa_{ij} = \frac{p_1 - p_2}{1 - p_2}$$

其中，p_1 为两个分类器取得一致的概率，p_2 为两个分类器**偶然**取得一致的概率，它们可由如下公式进行**估算**：

$$p_1 = \frac{a + d}{N}$$

$$p_2 = \frac{(a + b)(a + c) + (c + d)(b + d)}{N^2}$$

通常情况下，κ_{ij} 非负，仅在 h_i, h_j 达成一致的概率甚至低于偶然性的情况下取负值；κ_{ij} 越大，两个学习器越相似，当 $\kappa_{ij} = 1$ 时，两个学习器在数据集 S 上的表现完全一致，当 $\kappa_{ij} = 0$ 时，两个学习器在数据集 S 上的表现偶然达成一致。

κ **-误差图**是一个点云图，可直观地展示用于集成的个体学习器的准确性和多样性。图中每一个点代表一对学习器，也就是一共有 $T(T-1)/2$ 个点，每个点的横坐标表示这两个学习器的 κ -统计量，纵坐标表示这两个学习器的平均错误率。因此，点云越低，个体学习器越准确；点云越靠近右边，个体学习器的多样性越好。

- **多样性增强**

为了增强个体学习器的多样性，我们可以从样本、输入、输出、算法等几个方面来进行操作。

(1) 样本扰动：基于初始数据集，生成不同的训练集，再利用不同的训练集训练出不同的个体学习器，比如，Bagging 中的自助采样，AdaBoost 中的序列采样。

有些基学习器，比如决策树、神经网络等，训练样本稍加变化就会产生显著变动，因此数据样本扰动对于这些不稳定基学习器的多样性很有效；但那些对数据样本扰动不敏感的基学习器，即稳定基学习器 (stable base learner)，比如线性学习器、支持向量机、朴素贝叶斯、 k 近邻学习器等，我们则需使用其他的方法来增强多样性。

(2) 输入属性扰动

类似于特征工程，我们可以基于输入空间产生不同的特征子空间，再在不同的特征子空间中训练基学习器，比如随机子空间 (random subspace) 法。但若数据只包含少量属性，或者冗余属性很少，则不宜使用输入属性扰动法。

(3) 输出表示扰动

顾名思义就是对输出进行变动以增强多样性，比如，翻转法 (Flipping Output) 随机改变一些训练样本的标记；也可以对输出表示进行转化，如"输出调制法"(Output Smearing) 将分类输出转化为回归输出后构建个体学习器；还可将原任务拆解为多个可同时求解的子任务，如 ECOC 法利用纠错输出码将多分类任务拆解为一系列二分类任务来训练基学习器。

(4) 算法参数扰动

设置不同的基学习算法参数，比如神经网络的隐层神经元数、初始连接权值等。交叉验证实际上就是在选择最好的一组参数，而我们这里则考虑了多组算法参数。

1.2 集成策略

(1) 对于回归问题，最常见的策略是平均法：

- 简单平均法： $H(\boldsymbol{x}) = \dfrac{1}{T}\sum_{i=1}^{T} h_i(\boldsymbol{x})$
- 加权平均法： $H(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x}), w_i \geqslant 0, \sum_{i=1}^{T} w_i = 1$

加权平均法未必一定优于简单平均法，一般地，个体学习器性能相差较大时宜使用加权平均法，而个体学习器性能相近时宜使用简单平均法。此外，一般权重 w_i 非负才能确保集成性能优于单一最佳个体学习器。

贝叶斯模型平均 (Bayes model averaging, BMA) 基于后验概率为不同模型赋予权重，可视为加权平均法的一种特殊实现。

(2) 对于分类问题，常用策略是投票法。投票也分软投票 (soft voting) 和硬投票 (hard voting)。对于某个输入，若个体学习器输出的是各个类标记的概率，此时为软投票；若个体学习器确定地输出某个类标记，即在该标记下输出 1 ，在其他标记下输出 0 ，此时为硬投票。

绝对多数投票法 (majority voting) 就是输出得票超过半数的类标记，否则，拒绝预测。若一定要给出一个预测，则输出得票最多的标记，即相对多数投票法 (plurality voting)。和加权平均法类似，还有加权投票法 (weighted voting)。

(3) 学习法

学习法就是通过学习的方式来完成个体学习器的集成，典型代表是 Stacking。此时，个体学习器又称初级学习器，而用于集成的学习器称为次级学习器或元学习器 (meta-learner)。

Stacking 先基于初始样本集训练出个体学习器；接着，基于初始样本集和个体学习器生成新的用于训练次级学习器的训练集，该训练集的输入 $\boldsymbol{z}_i = (h_1(\boldsymbol{x}_i), \cdots, h_T(\boldsymbol{x}_i))^T$ ，对应输出为 $\boldsymbol{x}_i$ 的标记 y_i ，即 $(\boldsymbol{z}_i, y_i)$ ；最后，基于新的训练集 $\{(\boldsymbol{z}_i, y_i)\}_{i=1}^N$ 训练次级学习器，并用于个体学习器的集成。此外，考虑到直接使用个体学习器的训练集产生次级学习器的训练集所带来的过拟合风险，我们一般会使用类似于交叉验证或留一法这样的方式，用训练初级学习器未使用的样本来产生次级学习器的训练样本。

Stacking 与 BMA 的比较：理论上，若目标概念 c_{obj} 恰在当前考虑的模型中，且数据噪声很少，则 BMA 不差于 Stacking；然而，现实中无法确保 c_{obj} 一定在当前考虑的模型中，甚至可能难以用当前考虑的模型来进行近似，且 BMA 对模型近似误差非常敏感，因此 Stacking 通常优于 BMA，其鲁棒性更好。

根据个体学习器的生成方式，集成学习方法大致可分为如下两类：(1) 个体学习器间存在强依赖关系、必须串行生成的**序列化方法**，如，Boosting；(2) 个体学习器间不存在强依赖关系，可同时生成的**并行化方法**，如 Bagging 和随机森林 (Random forest)。下面两章，我们再具体介绍这两类典型的算法。

2 Bagging 与随机森林

2.1 Bagging

为了得到"好而不同"的基学习器，Bagging 对原始样本集 $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ 进行自主采样 (bootstrap sampling, 也就是有放回采样)，得到 T 个样本容量为 N 的采样集 S_t ，然后基于每个采样集训练出一个基学习器，再将这些基学习器进行集成。

可以看到，Bagging 基学习器的多样性策略为样本扰动。从偏差-方差分解的角度看，Bagging 主要关注降低方差，即通过集成多个学习器来降低同样大小训练集的变动所导致的学习性能的变化，从而提高泛化性能，所以它在不剪枝决策树、神经网络等易受样本扰动的学习器上效用更为明显。Bagging 的结合策略一般是简单投票法或简单平均法。

由自主采样的性质，每个采样集 S_t 只使用了原始样本集 S 中 $1/e \approx 63.2\%$ 的样本，剩下越 36.8% 的样本可用作验证集来对泛化性能进行包外估计 (out-of-bag estimate)。输入 $\boldsymbol{x}$ 的包外预测 $H_{oob}(\boldsymbol{x})$ 为：

$$H_{oob}(\boldsymbol{x}) = \arg\max_{y \in \mathbb{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y) \mathbb{I}(\boldsymbol{x} \neq S_t)$$

即，先集成那些未使用 $\boldsymbol{x}$ 训练的基学习器，然后输出其对 $\boldsymbol{x}$ 的预测。由此可得 Bagging 泛化误差的包外估计：

$$\hat{R}_S(H_{oob}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(H_{oob}(\boldsymbol{x}_i) \neq y_i)$$

此外，包外样本还有许多其他用途，如，辅助决策树进行剪枝；估计决策树中各结点的后验概率以辅助对零训练样本节点的处理；辅助神经网络的早期停止以减小过拟合的风险等

2.2 随机森林 (Random forest, RF)

随机森林是 Bagging 的一个扩展变体，相比 Bagging，其特别之处在于：(1) 基学习器为决策树，而 Bagging 中没有具体指定；(2) 在 Bagging 样本扰动的基础上加入了输入属性扰动。具体而言，在构建基决策树时，每次选择划分属性前会先随机生成一个包含 k 个属性的属性子集，再在其中选择最优的属性作为划分属性。可见，$k = d$ 时，基决策树的构建与传统决策树相同；$k = 1$ 时，随机选择一个属

性用于划分；一般地，取 $k = \log_2 d$ **。**

相比 Bagging，随机森林因引入输入属性扰动，其基决策树的性能往往有所降低，但多样性却增强了，因此随着基决策树数目的增加，随机森林通常会收敛到更低的泛化误差。此外，因为每次选择划分属性时只需考察一个随机的属性子集，即 k 个属性而不是全部的 d 个属性，所以其训练效率常优于 Bagging。总而言之，随机森林简单、容易实现，计算开销小，在很多现实任务中展现出强大的性能，被誉为"代表集成学习技术水平的方法"。

3 Boosting

Boosting 一族算法的思路如下：先从初始训练集训练出一个基学习器，再根据基学习器的表现对训练样本分布进行调整，使得先前基学习器做错的训练样本在后续受到更多关注，然后基于调整后的样本分布来训练下一个基学习器；如此重复进行，训练 T 轮得到 T 个基学习器，最后将这 T 个基学习器进行加权结合。

对 Boosting，我们说明如下：

1. 可以看到，**和 Bagging 一样，Boosting 基学习器的多样性策略也是样本扰动，但不同的是，从偏差-方差分解的角度看，Boosting 主要关注降低偏差，即提高对样本的拟合程度**：相比上一轮正确分类的样本，上一轮错误分类的样本在本轮中会被 Boosting 给予相对更多的关注，这使得本轮训练得到的基学习器会更偏向于将上一轮错误分类的样本正确分类，当然，相对地，这也可能会使上一轮正确分类的样本被本轮的学习器错误分类，为此，我们通过集成取长补短，从而令 Boosting 能基于泛化性能相当弱的学习器构建出很强的集成。
2. **Bagging 中各基学习器的训练相互独立，可并行生成；而 Boosting 的各基学习器串行生成。**
3. 有两种方法实现"对上一轮错误分类的样本给予更多的关注"：(1) 重赋权法 (re-weighting)；(2) 重采样法 (re-sampling)。重赋权法根据前一轮学习器的预测结果对每个训练样本重新赋一个权值 (比如，AdaBoost 中的 $\boldsymbol{w}_t$)，而对无法接受带权样本的基学习算法，则可采用重采样法根据前一轮训练得到的样本分布 (比如，AdaBoost 中的 $\boldsymbol{w}_t$) 对训练集进行重采样。

一般地，这两种方法没有显著的优劣差别，但重赋权法却可能出现训练轮数达不到预设的 T 轮即停止的情况。以下文中的 AdaBoost 为例，第 t 轮训练得到的基学习器 h_t 的加权经验误差 e_t 一定小于 0.5，但我们却无法保证没有经过 $\boldsymbol{w}_t$ 加权的经验误差 $e(h_t)$ 小于 0.5，即我们可能得到比随机猜测更差的学习器 h_t，而且此时我们还不能对 h_t 取反以使 $e(h_t) < 0.5$，因为 h_t 的目的是为了使上一轮错误分类的样本被正确分类，若取反则无法达到这一目的，会使得各基学习器对某些样本均预测正确而另一些样本均预测错误，即**基学习器的多样性不足，从而导致集成的效果比较差。而若** $e(h_t) > 0.5$**，即基学习器** h_t **的准确度不足，我们又无法将其用于集成。**此时，对于重赋权法，训练停止，而无法达到预设的 T 轮训练的要求；但对于重采样法，我们可以基于样本分布 $\boldsymbol{w}_t$ 再次进行重采样，直到重采样的样本使训练出来的学习器的经验误差 $e(h_t) < 0.5$，然后继续进行下一轮的训练，直到完成 T 轮训练。

3.1 AdaBoost

3.1.1 AdaBoost 算法

AdaBoost 针对二分类问题，训练集为 $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$，第 t 轮 (共 T 轮) 训练时的样本权重 $\boldsymbol{w}_t = (w_{t1}, \cdots, w_{tN})^T$，$w_{ti}$ **本质上给出了第** i **个样本点预测错误时的损失。我们不妨令** $\sum_{i=1}^N w_{ti} = 1$**，初始权重** $\boldsymbol{w}_1 = (1/N, \cdots, 1/N)^T$**，则权重** w_{ti} **可赋予概率的含义，即将** S **中的每个样本点看作一个事件，则第** i **个事件** $(\boldsymbol{x}_i, y_i)$ **出现的概率为** w_{ti}**，因此我们也称** $\boldsymbol{w}_t$ **为第** t **轮的样本分布。**可以看到，$\boldsymbol{w}_t$ **是一个离散的分布**，若取 $0-1$ 损失，则第 t 轮基训练器 h_t 在**样本分布** $\boldsymbol{w}_t$ 上的**期望风险** (注意不是经验风险) 为：

$$\sum_{i=1}^N w_{ti} \mathbb{I}(h_t(\boldsymbol{x}_i) \neq y_i) = \frac{1}{N} \sum_{i=1}^N (w_{ti} N) \mathbb{I}(h_t(\boldsymbol{x}_i) \neq y_i)$$

上式也是 h_t 在样本 S 上的**加权经验风险 (对应概率分布为 $(\boldsymbol{x}, y)$ 的联合分布)**，第 i 个样本点损失的权重为 $w_{ti}N$ 。**注意，这里所说的样本分布 $\boldsymbol{w}_t$ 和随机变量 $(\boldsymbol{x}, y)$ 的联合分布没有关系，并不是对 $(\boldsymbol{x}, y)$ 联合分布的近似，训练集 $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ 中可能存在两个值完全相等的样本点，记它们的值为 $(\boldsymbol{a}, b)$ ，但这两个点从样本分布的角度看并不是同一个事件，它们分别有属于自己的概率，这个概率与 $(\boldsymbol{a}, b)$ 在 S 中出现的频率无关。**此外，不要误认为 $\{(\boldsymbol{x}_i, w_{ti}y_i)\}_{i=1}^N$ 是第 t 轮的训练集，训练集仍然是 S 。

记基学习器的学习算法为 $\mathcal{L}$ ，第 t 轮学得的基学习器为 $h_t(\boldsymbol{x}) = \mathcal{L}(S, \boldsymbol{w}_t) : \mathbb{X} \to \{-1, +1\}$ ，学习器的权重为 α_t ，所有基学习器的线性组合： $f(\boldsymbol{x}) = \sum_{t=1}^T \alpha_t h_t(\boldsymbol{x}) : \mathbb{X} \to \mathbb{R}$ ，**$f(\boldsymbol{x})$ 的符号决定了输入的类，其绝对值给出了分类的确信度**，则最终分类器 $H(\boldsymbol{x}) = \operatorname{sign}(f(\boldsymbol{x})) : \mathbb{X} \to \{-1, +1\}$ 。**注意，和 $\boldsymbol{w}_t$ 不同， $\sum_{t=1}^T \alpha_t \neq 1$ 。此外，AdaBoost 并没有指定基学习器的类型，也就是没有指定具体的学习算法 $\mathcal{L}$ 。**

可以看到，需要我们求解的有 $\{h_t(\boldsymbol{x})\}_{t=1}^T$ 、 $\{\boldsymbol{w}_t\}_{t=1}^T$ 和 $\{\alpha_t\}_{t=1}^T$ 。其中，基学习器 $h_t(\boldsymbol{x})$ 由具体的学习算法 $\mathcal{L}$ 、样本 S 和样本权重 $\boldsymbol{w}_t$ 确定，而学习算法 $\mathcal{L}$ 具体给定，样本 S 已知，因此关键在于样本权重 $\boldsymbol{w}_t$ 和学习器权重 α_t 的计算。由下文可知，第 t 轮的学习器权重 α_t 依赖于学习器 $h_t(\boldsymbol{x})$ 的加权 (这个权指 $\boldsymbol{w}_t$) 经验误差 e_t ； 而第 t 轮的样本权重 $\boldsymbol{w}_t$ 则依赖于上一轮，也就是第 $t-1$ 轮的样本权重 $\boldsymbol{w}_{t-1}$ 和学习器权重 α_{t-1} 。

对第 t 轮训练：

1. 根据样本集 S 和样本权重 $\boldsymbol{w}_t$ 学得第 t 轮基学习器 $h_t = \mathcal{L}(S, \boldsymbol{w}_t)$ ，即：

$$h_t = \arg\min_h \frac{1}{N} \sum_{i=1}^N w_{ti} \mathbb{I}(h(\boldsymbol{x}_i) \neq y_i)$$

2. 计算基学习器 $h_t(\boldsymbol{x})$ 的加权分类误差率：

$$\begin{aligned} e_t &= \mathbb{P}\{h_t(\boldsymbol{x}_i) \neq y_i, i = 1, \cdots, N\} \\ &= \sum_{i=1}^N w_{ti} \mathbb{I}(h_t(\boldsymbol{x}_i) \neq y_i) \end{aligned}$$

此外，基学习器 h_t 的未加权分类错误率应满足小于 0.5 这一条件，即正常的分类错误率 $e(h_t) = \sum_{i=1}^N \mathbb{I}(h_t(\boldsymbol{x}_i) \neq y_i)/N < 0.5$ ； 否则 break。

3. 根据加权分类误差率 e_t 计算基学习器 $h_t(\boldsymbol{x})$ 的权重：

$$\alpha_t = \frac{1}{2} \ln \frac{1 - e_t}{e_t}$$

可以看到，**加权分类误差率越小的学习器在最终分类器中的作用越大。**

4. 根据本轮的样本权重 w_t 、学习器权重 α_t 以及学习器在样本上的表现，输出下一轮的样本权重：

$$\begin{aligned} w_{(t+1)i} &= \frac{w_{ti} \exp\left(-\alpha_t y_i h_t(\boldsymbol{x}_i)\right)}{Z_t} \\ &= \begin{cases} \dfrac{w_{ti} \exp\left(-\alpha_t\right)}{Z_t}, & h_t(\boldsymbol{x}_i) = y_i \\ \dfrac{w_{ti} \exp\left(\alpha_t\right)}{Z_t}, & h_t(\boldsymbol{x}_i) \neq y_i \end{cases} \quad i = 1, \cdots, N \end{aligned}$$

其中，分母 $Z_t = \sum_{i=1}^N w_{ti} \exp\left(-\alpha_t y_i h_t(\boldsymbol{x}_i)\right)$ 为样本权重的归一化因子，这使得样本权重是一个概率分布。可以看到，**本轮错误预测的样本在下一轮权重会降低，而本轮正确预测的样本在下一轮权重会升高。**

最后，我们给出 AdaBoost 训练误差分析，可以证明：

$$e(H)=\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(H(\boldsymbol{x}_i)\neq y_i)\leqslant\frac{1}{N}\sum_{i=1}^{N}\exp\left(-y_i f(\boldsymbol{x}_i)\right)=\prod_{t=1}^{T}Z_t$$
$$=\prod_{t=1}^{T}\left(2\sqrt{e_t(1-e_t)}\right)\leqslant\exp\left(-2\sum_{t=1}^{T}(\frac{1}{2}-e_t)^2\right)$$

若存在一个训练误差界 e^*，使得对所有 t 有 $e_t\leqslant e^*<\frac{1}{2}$，则：

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(H(\boldsymbol{x}_i)\neq y_i)\leqslant\exp\left(-2\sum_{t=1}^{T}\left(\frac{1}{2}-e_t\right)^2\right)\leqslant\exp\left(-2T\left(\frac{1}{2}-e^*\right)^2\right)$$

可以看到，**此时 AdaBoost 的训练误差以指数速率下降。**

3.1.2 加性模型 (additive model) 推导 AdaBoost 算法

1. 加性模型 (additive model)：$f(\boldsymbol{x})=\sum_{j=1}^{T}\alpha_j h_j(\boldsymbol{x};\theta_j)$，其中 α_j 为模型系数，θ_j 为模型参数，学习的目标就是计算它们的值。前向分段算法 (forward stagewise algorithm) 实际上就是一种贪心算法，一共训练 T 轮，每一轮在前一轮的基础上只优化一个模型：给定损失函数 L 和训练集 $S=\{(\boldsymbol{x}_i,y_i)\}_{i=1}^{N}$，我们已知前 $t-1$ 轮训练所得结果为 $f_{t-1}(\boldsymbol{x})=\sum_{j=1}^{t-1}\alpha_j h_j(\boldsymbol{x};\theta_j)$，则第 t 轮要训练的模型为：$f_t(\boldsymbol{x})=f_{t-1}(\boldsymbol{x})+\alpha h_t(\boldsymbol{x};\theta)$，待定未知量只有 α,θ，它们使经验风险最小化：

$$(\alpha_t,\theta_t)=\min_{\alpha,\theta}\frac{1}{N}\sum_{i=1}^{N}L(f_t(\boldsymbol{x}_i),y_i)$$

---

2. **AdaBoost 的一种解释是：模型为加性模型、损失函数为指数函数、优化算法为前向分段算法的二分类学习方法。**二分类问题的原始损失函数为 $0-1$ 损失，而这里我们使用的是其一致性替代损失函数--指数损失函数 $L(\hat{y},y)=\exp\left(-\hat{y}y\right)$，所谓的一致性就是说使用这两种损失函数进行学习所得结果相同，但显然，指数损失函数连续可微，性质更好。接下来我们就来推导 AdaBoost。

首先，我们说明如下：模型或基学习器 $h_j:\mathbb{X}\to\{-1,+1\}$ 的类型给定，其中含有待定参数 θ_j；$f_t=\sum_{j=1}^{t}\alpha_j h_j(\boldsymbol{x};\theta_j):\mathbb{X}\to\mathbb{R}$。和前文类似，$f_t(\boldsymbol{x}_i)$ 的符号决定了 $\boldsymbol{x}_i$ 的类，其绝对值给出了分类的确信度。因此，若使用前 t 个基学习器 $f_t(\boldsymbol{x})$ 进行分类，所得分类器为 $H_t(\boldsymbol{x})=\operatorname{sign}(f_t(\boldsymbol{x}))$，但为了更精确地描述预测损失，我们取指数损失函数为 $L(f_t(\boldsymbol{x}_i),y_i)=\exp\left(-y_i f_t(\boldsymbol{x}_i)\right)$，$y_i\in\{-1,+1\}$，而不是 $L(H_t(\boldsymbol{x}_i),y_i)$。可以看到，当 $f(\boldsymbol{x}_i)$ 与 y_i 同号时说明分类正确，此时损失小于 1，而 $|f(\boldsymbol{x}_i)|$ 越大，分类正确的程度就越大，损失就越小；当 $f(\boldsymbol{x}_i)$ 与 y_i 异号时说明分类错误，此时损失大于 1，而 $|f(\boldsymbol{x}_i)|$ 越大，分类错误的程度就越大，损失就越大。

如前所述，第 t 轮的训练：

$$(\alpha_t,\theta_t)=\arg\min_{\alpha,\theta}\frac{1}{N}\sum_{i=1}^{N}L(f_t(\boldsymbol{x}_i),y_i)$$
$$=\arg\min_{\alpha,\theta}\sum_{i=1}^{N}L(f_{t-1}(\boldsymbol{x}_i)+\alpha h_t(\boldsymbol{x}_i;\theta),y_i)$$
$$=\arg\min_{\alpha,\theta}\sum_{i=1}^{N}\exp\left(-y_i f_{t-1}(\boldsymbol{x}_i)-y_i\alpha h_t(\boldsymbol{x}_i;\theta)\right)$$

记常量 $\exp\left(-y_i f_{t-1}(\boldsymbol{x}_i)\right)\triangleq\hat{w}_{ti}$，继续对上式进行恒等变形：

$$(\alpha_t, \theta_t) = \arg\min_{\alpha,\theta} \sum_{i=1}^{N} \hat{w}_{ti} \exp\left(-y_i \alpha h_t(\boldsymbol{x}_i;\theta)\right)$$

$$= \arg\min_{\alpha,\theta} \sum_{i=1}^{N} \hat{w}_{ti} \exp\left(-\alpha\right) \mathbb{I}[h_t(\boldsymbol{x}_i;\theta) = y_i] + \sum_{i=1}^{N} \hat{w}_{ti} \exp\left(\alpha\right) \mathbb{I}[h_t(\boldsymbol{x}_i;\theta) \neq y_i]$$

$$= \arg\min_{\alpha,\theta} \exp\left(-\alpha\right) \sum_{i=1}^{N} \hat{w}_{ti} (1 - \mathbb{I}[h_t(\boldsymbol{x}_i;\theta) \neq y_i]) + \exp\left(\alpha\right) \sum_{i=1}^{N} \hat{w}_{ti} \mathbb{I}[h_t(\boldsymbol{x}_i;\theta) \neq y_i]$$

$$= \arg\min_{\alpha,\theta} \exp\left(-\alpha\right) \sum_{i=1}^{N} \hat{w}_{ti} + \left(\exp\left(\alpha\right) - \exp\left(-\alpha\right)\right) \sum_{i=1}^{N} \hat{w}_{ti} \mathbb{I}[h_t(\boldsymbol{x}_i;\theta) \neq y_i]$$

$$= \arg\min_{\alpha} \left\{ \exp\left(-\alpha\right) \sum_{i=1}^{N} \hat{w}_{ti} + \min_{\theta} \left\{ \left(\exp\left(\alpha\right) - \exp\left(-\alpha\right)\right) \sum_{i=1}^{N} \hat{w}_{ti} \mathbb{I}[h_t(\boldsymbol{x}_i;\theta) \neq y_i] \right\} \right\}$$

$$= \arg\min_{\alpha} \left\{ \exp\left(-\alpha\right) \sum_{i=1}^{N} \hat{w}_{ti} + \left(\exp\left(\alpha\right) - \exp\left(-\alpha\right)\right) \min_{\theta} \left\{ \sum_{i=1}^{N} \hat{w}_{ti} \mathbb{I}[h_t(\boldsymbol{x}_i;\theta) \neq y_i] \right\} \right\}$$

上式倒数第二行到最后一行是因为系数 $\alpha > 0$，则 $\exp\left(\alpha\right) - \exp\left(-\alpha\right) > 0$，最小化保持，不会变成最大化。那么，我们有：

(1) 我们先讨论一下 $\hat{\boldsymbol{w}}_t$ 与 AdaBoost 中的权重 $\boldsymbol{w}_t$ 之间的关系。第一轮训练时，$\boldsymbol{w}_1 = (1/N, \cdots, 1/N)^T$，不妨取 $\hat{\boldsymbol{w}}_1 = (1, \cdots, 1)^T$，即两个向量满足 $\hat{\boldsymbol{w}}_1 = N\boldsymbol{w}_1$；又易知 $\hat{w}_{ti}$ 的递推公式：$\hat{w}_{(t+1)i} = \exp\left(-y_i \alpha_t h_t(\boldsymbol{x}_i;\theta_j)\right) \hat{w}_{ti}$，$w_{(t+1)i} = \exp\left(-\alpha_t y_i h_t(\boldsymbol{x}_i)\right) w_{ti} / Z_t$，我们记对角矩阵 $\Lambda_t = diag\{\exp\left(-\alpha_t y_i h_t(\boldsymbol{x}_i)\right)\}_{i=1}^{N}$，则 $\hat{\boldsymbol{w}}_{t+1} = \Lambda_t \hat{\boldsymbol{w}}_t$，$\boldsymbol{w}_{t+1} = \Lambda_t \boldsymbol{w}_t / Z_t$；由此，我们可推得：

$$\hat{\boldsymbol{w}}_{t+1} = \left(N \prod_{j=1}^{t} Z_j\right) \boldsymbol{w}_{t+1} = \text{const} \cdot \boldsymbol{w}_{t+1}$$

则对于参数 θ_t：

$$\theta_t = \arg\min_{\theta} \sum_{i=1}^{N} \hat{w}_{ti} \mathbb{I}[h_t(\boldsymbol{x}_i;\theta) \neq y_i]$$

$$= \arg\min_{\theta} \sum_{i=1}^{N} w_{ti} \mathbb{I}[h_t(\boldsymbol{x}_i;\theta) \neq y_i]$$

$$\Updownarrow$$

$$h_t = \mathcal{L}(S, \boldsymbol{w}_t)$$

(2) 对于系数 α_t：我们取 $\theta = \theta_t$，则

$$\alpha_t = \arg\min_{\alpha} \left\{ \exp\left(-\alpha\right) \sum_{i=1}^{N} \hat{w}_{ti} + \left(\exp\left(\alpha\right) - \exp\left(-\alpha\right)\right) \left(\sum_{i=1}^{N} \hat{w}_{ti} \mathbb{I}[h_t(\boldsymbol{x}_i;\theta_t) \neq y_i] \right) \right\}$$

求导法计算最值可得：

$$\alpha_t = \frac{1}{2}\ln\frac{1-e_t}{e_t}$$

$$\text{where}$$

$$e_t = \frac{\sum_{i=1}^N \hat{w}_{ti}\mathbb{I}[h_t(\boldsymbol{x}_i;\theta_t)\neq y_i]}{\sum_{i=1}^N \hat{w}_{ti}}$$

$$= \sum_{i=1}^N w_{ti}\mathbb{I}[h_t(\boldsymbol{x}_i)\neq y_i]$$

可以看到，**所得结果与 AdaBoost 一致。**

---

3. 细心的读者可能注意到了本小节与上一小节表述上的不同，上一小节介绍 AdaBoost 的时候，没有出现参数 θ_t ，我们直接说的是对基学习器 h_t 进行学习，而这里我们说的是对参数 θ_t 进行学习。二者表达的意思相同，因为基学习器 h_t 的类型都事先给定，我们要确定的就是里面的参数 θ_t ，因此这两种表达方式可以相互转化。但可以看到，因为二者的表达形式不同，这使得前者是一个**泛函问题**，优化的变量是一个函数 h (但并不是没有限制的，函数的形式给定)，我们是在一个**函数空间**内寻找最优解，而后者是一个普通的优化问题，我们是在一个**参数空间**内寻找最优解：

$$h_t = \arg\min_h \sum_{i=1}^N w_{ti}\mathbb{I}(h(\boldsymbol{x}_i)\neq y_i)$$

$$\theta_t = \arg\min_\theta \sum_{i=1}^N w_{ti}\mathbb{I}[h_t(\boldsymbol{x}_i;\theta)\neq y_i]$$

若在加性模型推导 AdaBoost 中采用基于泛函的表述，对于第 t 轮的训练：

$$(\alpha_t, h_t) = \arg\min_{\alpha,h}\sum_{i=1}^N L(f_{t-1}(\boldsymbol{x}_i)+\alpha h(\boldsymbol{x}_i), y_i)$$

只看右端的优化目标函数，变量为 $h(\boldsymbol{x}_1),\cdots,h(\boldsymbol{x}_N),\alpha$ ，那么我们能否通过对这 $N+1$ 个变量求导来求解上述优化问题呢？答案是不行，原因有二：(1) 样本点 $\boldsymbol{x}_i$ 并不具有任意性，因此我们只能得到 h_t 在每个样本点 $\boldsymbol{x}_i$, $i=1,\cdots,N$ 上的值 (不妨记为 $\hat{y}_i$) 而不是一个函数；(2) 此外， h 的形式并不是任意的，而是提前设定好的，即 h 属于一个函数空间，因此我们无法保证在这个函数空间中存在一个函数 h_t 使得 $h_t(\boldsymbol{x}_i)=\hat{y}_i$ 对所有的 $i=1,\cdots,N$ 都成立。

---

4. 西瓜书和论文《Additive logistic regression a statistical view of boosting》中在每轮学习时采用类似牛顿法推得了 AdaBoost。这里所说的牛顿法是指针对优化问题的牛顿法，而不是用于解方程的牛顿法。下面，我们给出有关细节。可以看到，**下面的方法借鉴了牛顿法的二阶泰勒展开近似步，只不过牛顿法接下来还会进行迭代，而这里并没有进行迭代了，XGBoost 就是采用的这种处理方法。**

第 t 轮训练要解决的问题：

$$(\alpha_t,\theta_t) = \arg\min_{\alpha,\theta}\sum_{i=1}^N \exp\left(-y_i f_{t-1}(\boldsymbol{x}_i) - y_i\alpha h_t(\boldsymbol{x}_i;\theta)\right)$$

若给定 $h_t(\boldsymbol{x}_i;\theta)$ ，则可推得：

$$\alpha_t = \frac{1}{2}\ln\frac{1-e_t}{e_t}$$

若给定 α，最小化 $\sum_{i=1}^{N}\exp\left(-y_i f_{t-1}(\boldsymbol{x}_i)-y_i\alpha h_t(\boldsymbol{x}_i;\theta)\right)=\sum_{i=1}^{N}\hat{w}_{ti}\exp\left(\alpha\right)\exp\left(-y_i h_t(\boldsymbol{x}_i;\theta)\right)$。对 $\exp\left(-y_i h_t(\boldsymbol{x}_i;\theta)\right)$ 在 0 处进行二阶泰勒展开，注意到 $y_i, h_t(\boldsymbol{x}_i;\theta)\in\{-1.+1\}$，则：

$$\begin{aligned}\exp\left(y_i h_t(\boldsymbol{x}_i;\theta)\right)&\approx 1+\left(-y_i h_t(\boldsymbol{x}_i;\theta)-0\right)+\frac{1}{2}\left(-y_i h_t(\boldsymbol{x}_i;\theta)-0\right)^2\\&=1-y_i h_t(\boldsymbol{x}_i;\theta)+\frac{1}{2}=\frac{3}{2}-y_i h_t(\boldsymbol{x}_i;\theta)\\\sum_{i=1}^{N}\hat{w}_{ti}\exp\left(\alpha\right)\exp\left(-y_i h_t(\boldsymbol{x}_i;\theta)\right)&\approx\sum_{i=1}^{N}\hat{w}_{ti}\exp\left(\alpha+\frac{3}{2}\right)\exp\left(-y_i h_t(\boldsymbol{x}_i;\theta)\right)\end{aligned}$$

则：

$$\min_{\theta}\sum_{i=1}^{N}\exp\left(-y_i f_{t-1}(\boldsymbol{x}_i)-y_i\alpha h_t(\boldsymbol{x}_i;\theta)\right)\approx\min_{\theta}\sum_{i=1}^{N}\hat{w}_{ti}\exp\left(-y_i h_t(\boldsymbol{x}_i;\theta)\right)$$

推导结束。

注意，**这里的 $\theta_t'=\arg\min_\theta\sum_{i=1}^{N}\hat{w}_{ti}\exp\left(-y_i h_t(\boldsymbol{x}_i;\theta)\right)$ 与前文所描述的 AdaBoost 算法中的 $\theta_t=\arg\min_\theta\sum_{i=1}^{N}\hat{w}_{ti}\mathbb{I}[h_t(\boldsymbol{x}_i;\theta)\neq y_i]$ 所得结果是不同的，θ_t' 是 θ_t 的近似。这与我们所说的 $0-1$ 损失与指数损失一致并不矛盾，因为一致性是针对期望风险最小化所得的决策函数而言的。事实上，这里给出了两个版本的 AdaBoost 算法，二者的区别在对 $(\alpha_t,\theta_t)=\arg\min_{\alpha,\theta}\sum_{i=1}^{N}\exp\left(-y_i f_{t-1}(\boldsymbol{x}_i)-y_i\alpha h_t(\boldsymbol{x}_i;\theta)\right)$ 的求解，前文所描述的 AdaBoost 采用精确解 $\arg\min_\theta\sum_{i=1}^{N}\hat{w}_{ti}\mathbb{I}[h_t(\boldsymbol{x}_i;\theta)\neq y_i]$；而这里的 AdaBoost 采用近似解 $\theta_t=\arg\min_\theta\sum_{i=1}^{N}\hat{w}_{ti}\exp\left(-y_i h_t(\boldsymbol{x}_i;\theta)\right)$，显然这会使两个版本所得的 α 不同。**

---

5. 加性模型从统计视角给出了 AdaBoost 的推导，但也仅仅只是 AdaBoost 的一种观察视角，并没有阐释 AdaBoost 的本质，无法解释“AdaBoost 在训练误差达到零 (应该是指集成学习器的未加权训练误差，而不是基学习器的加权训练误差，基学习器的学习能力有限，不可能出现训练误差达到零的情况) 后继续训练仍能提高泛化能力，但若一直训练下去，过拟合最终仍然会出现”这一现象。该现象可用间隔理论 (margin theory) 进行解释。众所周知，以支持向量机为代表的一大类统计学习方法都在试图最大化“最小间隔”，而这个间隔理论揭示：影响泛化能力的物理量实际上有两个，即“平均间隔”和“间隔方差”，若能最大化“平均间隔”同时最小化“间隔方差”，得到的学习器会更好！AdaBoost 在训练过程中随着轮数的增加，不仅使“平均间隔”增大，还同时使“间隔方差”变小。同时，这也意味着 AdaBoost 最终仍有可能发生过拟合，只不过很迟——当“平均间隔”已无法再增大、“间隔方差”也无法进一步减小时，可参见 周志华：Boosting学习理论的探索 —— 一个跨越30年的故事 。

> 个人的想法是：就算基学习器能将所有样本正确分类，我们也可以根据各样本点“正确程度”的不同，继续改变样本的分布进行训练。比如，若第 t 轮得到的基分类器 $h_t(\boldsymbol{x})=\text{sign}(f_t(\boldsymbol{x}))$ 的训练误差为 0，但 $f_t(\boldsymbol{x})$ 却可以衡量样本点的“正确程度”，绝对值越大的样本点“正确程度”越高，而对“正确程度”低的样本点，我们可以给予更多的关注，这也就改变了样本分布，从而可以继续训练下去。这个“正确程度”在 SVM 中就体现为间隔，相比感知机只要能正确分类训练样本算法就停止，SVM 还需最大化“最小间隔”，因此泛化性能会更好。

3.2 Gradient Boosting 和 XGBoost

参见笔记《机器学习8b-Gradient Boosting 和 XGBoost》。

4 附录 - $0-1$ 损失与指数损失一致性证明

假设随机变量 $(\boldsymbol{x},y)$ 的条件概率分布为 $P(y|\boldsymbol{x})$，损失函数 L，则决策函数 $c_{obj}:\mathbb{X}\to\mathbb{Y}$ 应该使期望风险最小化，即为求解如下的**泛函问题 (即优化问题的变量是一个函数)**：

$$
\begin{aligned}
c_{obj} &= \arg\min_{c} \mathbb{E}_{(\boldsymbol{x},y)\sim P(\boldsymbol{x},y)}\{L(c(\boldsymbol{x}),y)\} \\
&= \arg\min_{c} \int_{\boldsymbol{x}\in\mathbb{X},y\in\mathbb{Y}} L(c(\boldsymbol{x}),y)P(\boldsymbol{x},y)\mathrm{d}\boldsymbol{x}\mathrm{d}y \\
&= \arg\min_{c} \int_{\boldsymbol{x}\in\mathbb{X},y\in\mathbb{Y}} L(c(\boldsymbol{x}),y)P(y|\boldsymbol{x})P(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\mathrm{d}y \\
&= \arg\min_{c} \mathbb{E}_{\boldsymbol{x}\sim P(\boldsymbol{x})}\{L(c(\boldsymbol{x}),y)P(y|\boldsymbol{x})\}
\end{aligned}
$$

其中，映射 $c:\mathbb{X}\to\mathbb{Y}$，注意上式返回的是一个函数 c_{obj} 而不是某个点 $\boldsymbol{x}$ 的函数值 $c_{obj}(\boldsymbol{x})$，是一个**泛函问题**。不妨令 $\mathbb{X}=\{\boldsymbol{q}_1,\boldsymbol{q}_2\cdots\}$，我们要求函数 c_{obj}，也就是计算各 $c_{obj}(\boldsymbol{q}_i)\in\mathbb{Y}$ 的值，记变量 $o_i\in\mathbb{Y}$：

$$
\begin{aligned}
\{c_{obj}(\boldsymbol{q}_1),c_{obj}(\boldsymbol{q}_2),\cdots\} = \arg\min_{o_1,o_2,\cdots} \\
\int_{\mathbb{Y}} L(o_1,y)P(y|\boldsymbol{q}_1)P(\boldsymbol{q}_1)\mathrm{d}y \\
+\int_{\mathbb{Y}} L(o_2,y)P(y|\boldsymbol{q}_2)P(\boldsymbol{q}_2)\mathrm{d}y \\
+\cdots
\end{aligned}
$$

上式第 i 个被加项只是 o_i 的函数，被加项相互之间没有影响，因此上式等价于：

$$
\begin{aligned}
c_{obj}(\boldsymbol{q}_1) &= \arg\min_{o_1\in\mathbb{Y}} \int_{\mathbb{Y}} L(o_1,y)P(y|\boldsymbol{q}_1)P(\boldsymbol{q}_1)\mathrm{d}y = \arg\min_{o_1\in\mathbb{Y}} \int_{\mathbb{Y}} L(o_1,y)P(y|\boldsymbol{q}_1)\mathrm{d}y \\
c_{obj}(\boldsymbol{q}_2) &= \arg\min_{o_2\in\mathbb{Y}} \int_{\mathbb{Y}} L(o_2,y)P(y|\boldsymbol{q}_2)P(\boldsymbol{q}_2)\mathrm{d}y = \arg\min_{o_2\in\mathbb{Y}} \int_{\mathbb{Y}} L(o_2,y)P(y|\boldsymbol{q}_2)\mathrm{d}y \\
&\cdots
\end{aligned}
$$

即 $c_{obj}(\boldsymbol{q}_i)=\arg\min_{o\in\mathbb{Y}}\int_{\mathbb{Y}} L(o,y)P(y|\boldsymbol{q}_i)\mathrm{d}y$，由 $\boldsymbol{q}_i$ 的任意性：

$$
\begin{aligned}
c_{obj}(\boldsymbol{x}) &= \arg\min_{o\in\mathbb{Y}} \int_{\mathbb{Y}} L(o,y)P(y|\boldsymbol{x})\mathrm{d}y \\
&= \arg\min_{o\in\mathbb{Y}} \mathbb{E}_{P(y|\boldsymbol{x})}\{L(o,y)\}
\end{aligned}
$$

进一步，若损失函数 L 取 $0-1$ 损失，则可推得决策函数：

$$
c_{obj}(\boldsymbol{x}) = \arg\min_{y\in\mathbb{Y}} P(y|\boldsymbol{x})
$$

上述即为贝叶斯决策论中的内容。

接下来我们来证明，**对于二分类问题，指数损失函数是 $0-1$ 损失函数的一致性 (consistent) 替代损失函数，即使用指数损失函数和使用 $0-1$ 损失函数进行期望风险最小化所得决策函数相同，也就是学习所得结果相同。**指数损失函数：

$$
L(\hat{y},y)=\exp\left(-y\hat{y}\right)
$$

则决策函数：

$$
\begin{aligned}
c_{obj}(\boldsymbol{x}) &= \arg\min_{o\in\{-1,+1\}} \int_{\mathbb{Y}} L(o,y)P(y|\boldsymbol{x})\mathrm{d}y \\
&= \arg\min_{o\in\{-1,+1\}} \exp(o)P(-1|\boldsymbol{x}) + \exp(-o)P(+1|\boldsymbol{x}) \\
&= \arg\min_{o\in\{-1,+1\}} \exp(o)P(-1|\boldsymbol{x}) + \frac{P(+1|\boldsymbol{x})}{\exp(o)} \\
&= \begin{cases} -1, & P(-1|\boldsymbol{x}) > P(+1|\boldsymbol{x}) \\ +1, & P(-1|\boldsymbol{x}) < P(+1|\boldsymbol{x}) \end{cases} \\
&= \arg\min_{y\in\{-1,+1\}} P(y|\boldsymbol{x})
\end{aligned}
$$