机器学习（西瓜书）

注　解

（第 8 章　集成学习）

https://blog.csdn.net/jbb0523

前言

经常听人说南大周老师所著的《机器学习》（以下统称为西瓜书）是一本入门教材，是一本科普性质的教科书。在该书第十次印刷之际，周老师在“如何使用本书”中也提到“这是一本入门级教科书”。然而，本人读起来却感觉该书远不止“科普”“入门”那么简单，书中的很多公式需要思考良久方能推导，很多概念需要反复咀嚼才能消化。边读边想着是不是应该将自己学习时遇到的一些知识难点的解析分享出来，以帮助更多的人入门。自己的确也随手做过一些笔记，但由于怀疑这仅是自己的个别现象，毕竟读书期间，思考更多的是如何使用单片机、DSP、ARM、FGPA 等，而这些基本是不需要推导任何公式的，因此作罢。偶然间在周老师的新浪微博看到如下对话：

此时方知，可能“读不懂”并不是个别现象。因此决定写一本“西瓜书注解”或者称为“西瓜书读书笔记”，对自己研读西瓜书时遇到的“台阶”进行解释和推导，以帮助更多的人能够更快地进入到这个领域。另外，近期越来越强地意识到，扎扎实实地推导一些基础算法的公式，无论是对于理解算法本身机理还是进行学术研究，都是非常有必要的。

自己会根据个人学习进度和研究需要按章发布，不知道能不能坚持写完，加油！

毕竟自己也是一名初学者，所以可能一些概念解释并不完整、一些公式推导并不优美，甚至会存在错误，这是不可避免的，不接受漫骂，但欢迎将问题反馈给我，共同学习进步！

（网盘链接：https://pan.baidu.com/s/1QtEiNnk8jMzmbs0KPBN-_w）

第 8 章目录

第 8 章 集成学习

首先，我们来看一下西瓜书作者的谷歌学术主页（2018-11-18）：

标题	引用次数	年份
Top 10 algorithms in data mining X Wu, V Kumar, JR Quinlan, J Ghosh, Q Yang, H Motoda, GJ McLachlan, ... Knowledge and information systems 14 (1), 1-37	4149	2008
Ensembling neural networks: many could be better than all ZH Zhou, J Wu, W Tang Artificial intelligence 137 (1-2), 239-263	1773	2002
ML-KNN: A lazy learning approach to multi-label learning ML Zhang, ZH Zhou Pattern recognition 40 (7), 2038-2048	1614	2007
Exploratory undersampling for class-imbalance learning XY Liu, J Wu, ZH Zhou IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics 39 (2), 539-550	995	2009
Ensemble Methods: Foundations and Algorithms ZH Zhou Chapman & Hall/CRC Press	939	2012
A Review on Multi-Label Learning Algorithms ML Zhang, ZH Zhou IEEE Trans. Knowledge and Data Engineering 26 (8), 1819-1837	926	2014
Automatic age estimation based on facial aging patterns X Geng, ZH Zhou, K Smith-Miles IEEE Trans. Pattern Analysis and Machine Intelligence 29 (12), 2234-2240	835	2007

引用次数前 5 名中，除第 3 名“ML-KNN: A lazy learning approach to multi-label learning”与集成学习(ensemble learning)无关以外，其余均与集成学习有直接关系。其中：

第 1 名的“Top 10 algorithms in data mining”是在 ICDM’06 中投票选出的数据挖掘十大算法，每个提名算法均由业内专家代表去阐述，然后进行投票，其中最终得票排名第 7 位的“Adaboost”即由西瓜书作者作为代表进行阐述；第 2 名的“Ensembing neural networks: many could be better than all”是西瓜书作者最早的代表作，该文催生了基于优化的集成修剪(Ensemble pruning)技术；第 4 名的“Exploratory undersampling for class-imbalance learning”是以集成学习技术解决类别不平衡问题；第 5 名的“Ensemble Methods: Foundations and Algorithms”则是西瓜书作者所著的集成学习专著。

因此，虽然周老师现在的研究方向已涉及到了机器学习的方方面面，但若将集成学习看作是其学术之路的源头，实际也不为过。

所以，认真享受西瓜书作者最拿手的本章内容吧^_^

8.1 个体与集成

本节内容通俗易懂，几乎不需要什么注解。

基学习器(base learner)的概念在论文中经常出现，可留意一下；另外，本节提到的投票法有两种，除了本节的多数投票(majority voting)，还有概率投票(probability voting)，这两点在 8.4 节中均会提及，即硬投票和软投票。

1、式(8.1)的解释

注意到$f(\boldsymbol{x})$为$\boldsymbol{x}$的真实标记，而$h_i(\boldsymbol{x})$为基分类器h_i对$\boldsymbol{x}$的预测标记，因此$h_i(\boldsymbol{x}) \neq f(\boldsymbol{x})$

表示预测标记与真实标记不相同的情况，即预测出错，$P(h_i(\boldsymbol{x}) \neq f(\boldsymbol{x}))$即为错误率。

2、式(8.2)的解释

注意到当前仅针对二分类问题$y \in \{-1, +1\}$，即预测标记$h_i(\boldsymbol{x}) \in \{-1, +1\}$。因此，若$T$个基分类器中预测输出$+1$占半数以上，则$\sum_{i=1}^{T} h_i(\boldsymbol{x}) > 0$；反之，若$T$个基分类器中预测输出$-1$占半数以上，则$\sum_{i=1}^{T} h_i(\boldsymbol{x}) < 0$。函数$\mathrm{sign}(\cdot)$为符号函数，在$\cdot < 0$时取值为$-1$，在$\cdot > 0$时取值为$+1$，即本式的函数$H(\boldsymbol{x})$与多数基分类器输出一致，也就是多数投票。

3、式(8.3)的推导

本式推导基于 Hoeffding 不等式。首先将 P192 习题 8.1 的式(8.43)变形如下：

$$P\left(H(T) \leqslant \lfloor T/2 \rfloor\right) = \sum_{k=0}^{\lfloor T/2 \rfloor} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k}$$

从式(8.43)到上式的符号变化包括$n \to T, k \to \lfloor T/2 \rfloor, i \to k, p \to (1-\epsilon)$。相应的，对于接下来的式(8.44)，则对$\delta > 0, \lfloor T/2 \rfloor = (1-\epsilon-\delta)T$，有 Hodeffding 不等式

$$P\left(H(T) \leqslant (1-\epsilon-\delta)T\right) = e^{-2\delta^2 T}$$

对于$\lfloor T/2 \rfloor = (1-\epsilon-\delta)T$，整理变形得$\delta$表达式

$$\delta = 1 - \epsilon - \frac{\lfloor T/2 \rfloor}{T}$$

注意到集成学习要求基分类器泛化性能优于随机猜测的学习器，例如在二分类问题上精度略高于 50%的分类器，即$\epsilon < 0.5$，因此$1-\epsilon > \frac{1}{2}$；而$\frac{\lfloor T/2 \rfloor}{T} \leqslant \frac{1}{2}$（符号$\lfloor \cdot \rfloor$表示向下取整，“$=$”在$T$为偶数时成立，若$T$为奇数，则一定是“$<$”）。因此

$$\delta = 1 - \epsilon - \frac{\lfloor T/2 \rfloor}{T} \geqslant 1 - \epsilon - \frac{1}{2} > 0$$

函数$f(x) = e^{-Cx^2}$为单调递减函数，其中$C > 0$为常数，因此

$$P\left(H(T) \leqslant \lfloor T/2 \rfloor\right) = e^{-2\delta^2 T} \leqslant e^{-2\left(1-\epsilon-\frac{1}{2}\right)^2 T} = e^{-\frac{T}{2}(1-2\epsilon)^2}$$

而根据式(8.2)中$H(\boldsymbol{x})$的含义，$H(T) \leqslant \lfloor T/2 \rfloor$（即$T$个基分类器中预测正确的个数不超过$\lfloor T/2 \rfloor$，也就是预测错误）与$H(\boldsymbol{x}) \neq f(\boldsymbol{x})$等价，推导完毕！

注意，上式推导中要保证$\delta > 0$，如此才能保证δ与δ^2符号一致，不等式缩放也一致；习题 8.1 中介绍的 Hoeffding 不等式仅为伯努力随机变量特例，参见 CSDN 博客“机器学习数学原理（8）——霍夫丁不等式”（https://blog.csdn.net/z_x_1996/article/details/73564926）。

8.2 Boosting

注意 8.1 节最后一段提到：根据个体学习器的生成方式，目前的集成学习方法大致可分为两大类，即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，以及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。

本节 Boosting 为前者的代表，Adaboost 又是 Boosting 族算法的代表。

1、式(8.4)的解释

注意到$h_t(\boldsymbol{x}) \in \{-1,+1\}$，加权系数$\alpha_t$由后面的式(8.11)给出（$T$个系数$\alpha_t$仅与自身对应的基学习器错误率$\epsilon_t$有关，相互之间无必然联系；由于要求$\epsilon_t < 0.5$，因此$\alpha_t > 0$），所得线性加权结果$H(\boldsymbol{x})$就是一个实数，可能大于零，也可能小于零，取值范围无法确定。

2、式(8.5)的解释

(1)先考虑指数损失函数$e^{-f(\boldsymbol{x})H(\boldsymbol{x})}$的含义（参见 P131 图 6.5）：$f$为真实函数，对于样本$\boldsymbol{x}$来说，$f(\boldsymbol{x}) \in \{-1,+1\}$只能取$-1$和$+1$两个值，而$H(\boldsymbol{x})$是一个实数；

当$H(\boldsymbol{x})$的符号与$f(\boldsymbol{x})$一致时，$f(\boldsymbol{x})H(\boldsymbol{x}) > 0$，因此$e^{-f(\boldsymbol{x})H(\boldsymbol{x})} = e^{-|H(\boldsymbol{x})|} < 1$，且$|H(\boldsymbol{x})|$越大指数损失函数$e^{-f(\boldsymbol{x})H(\boldsymbol{x})}$越小（这很合理：此时$|H(\boldsymbol{x})|$越大意味着分类器本身对预测结果的信心越大，损失应该越小；若$|H(\boldsymbol{x})|$在零附近，虽然预测正确，但表示分类器本身对预测结果信心很小，损失应该较大）；

当$H(\boldsymbol{x})$的符号与$f(\boldsymbol{x})$不一致时，$f(\boldsymbol{x})H(\boldsymbol{x}) < 0$，因此$e^{-f(\boldsymbol{x})H(\boldsymbol{x})} = e^{|H(\boldsymbol{x})|} > 1$，且$|H(\boldsymbol{x})|$越大指数损失函数$e^{-f(\boldsymbol{x})H(\boldsymbol{x})}$越大（这很合理：此时$|H(\boldsymbol{x})|$越大意味着分类器本身对预测结果的信心越大，但预测结果是错的，因此损失应该越大；若$|H(\boldsymbol{x})|$在零附近，虽然预测错误，但表示分类器本身对预测结果信心很小，虽然错了，损失应该较小）；

对比 0/1 损失函数（参见 P131 图 6.5）：当$H(\boldsymbol{x})$的符号与$f(\boldsymbol{x})$一致时则损失为 0，当$H(\boldsymbol{x})$的符号与$f(\boldsymbol{x})$不一致时则损失为 1，而不管$H(\boldsymbol{x})$的大小。

(2)再解释符号$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\cdot]$的含义：$\mathcal{D}$为概率分布，可简单理解为在数据集$D$中进行一次随机抽样，每个样本被取到的概率；$\mathbb{E}[\cdot]$为经典的期望，则综合起来$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\cdot]$表示在概率分布$\mathcal{D}$上的期望，可简单理解为对数据集$D$以概率$\mathcal{D}$进行加权后的期望。

综上所述，若数据集D中样本$\boldsymbol{x}$的权值分布为$\mathcal{D}(\boldsymbol{x})$，则式(8.5)可写为：

$$\begin{aligned}\ell_{\exp}(H \mid \mathcal{D}) =& \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H(\boldsymbol{x})}] \\ =& \sum_{\boldsymbol{x}\in D} \mathcal{D}(\boldsymbol{x}) e^{-f(\boldsymbol{x})H(\boldsymbol{x})} \\ =& \sum_{\boldsymbol{x}\in D} \mathcal{D}(\boldsymbol{x}) \left(e^{-H(\boldsymbol{x})}\mathbb{I}(f(\boldsymbol{x})=1) + e^{H(\boldsymbol{x})}\mathbb{I}(f(\boldsymbol{x})=-1)\right)\end{aligned}$$

特别地，若针对任意样本$\boldsymbol{x}$，若分布$\mathcal{D}(\boldsymbol{x}) = \frac{1}{|D|}$，其中$|D|$为数据集$D$样本个数，则

$$\ell_{\exp}(H \mid \mathcal{D}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H(\boldsymbol{x})}] = \frac{1}{|D|}\sum_{\boldsymbol{x}\in D} e^{-f(\boldsymbol{x})H(\boldsymbol{x})}$$

而这就是在求传统平均值。

3、式(8.6)的推导

首先，当$f(\boldsymbol{x}) = 1$时$e^{-f(\boldsymbol{x})H(\boldsymbol{x})} = e^{-H(\boldsymbol{x})}$，当$f(\boldsymbol{x}) = -1$时$e^{-f(\boldsymbol{x})H(\boldsymbol{x})} = e^{H(\boldsymbol{x})}$；

其次，由以上分析可知式(8.5)包含$|D|$个不同的$e^{-f(\boldsymbol{x})H(\boldsymbol{x})}$，其中$\boldsymbol{x} \in D$；

最后，针对每个$H(\boldsymbol{x})$求导，只有对应的$e^{-f(\boldsymbol{x})H(\boldsymbol{x})}$会起作用，其余与$H(\boldsymbol{x})$无关的项求

导后等于零，若不理解本句话，将式(8.5)的解释中的式(8.5)改写为如下形式

$$\begin{aligned}\ell_{\exp}(H \mid \mathcal{D}) =& \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H(\boldsymbol{x})}] = \sum_{i=1}^{|D|} \mathcal{D}(\boldsymbol{x}_i) e^{-f(\boldsymbol{x}_i)H(\boldsymbol{x}_i)} \\ =& \sum_{i=1}^{|D|} \mathcal{D}(\boldsymbol{x}_i) \left(e^{-H(\boldsymbol{x}_i)}\mathbb{I}(f(\boldsymbol{x}_i)=1) + e^{H(\boldsymbol{x}_i)}\mathbb{I}(f(\boldsymbol{x}_i)=-1)\right)\end{aligned}$$

但这里有点不同的是，式(8.6)是为了推导出式(8.7)和式(8.8)的判定准则预测$\boldsymbol{x}$的类别，即无法得知$f(\boldsymbol{x})$具体等于1还是-1，只能知道$f(\boldsymbol{x})$为 1 的后验概率$P(f(\boldsymbol{x})=1 \mid \boldsymbol{x})$和为$-1$的后验概率$P(f(\boldsymbol{x})=-1 \mid \boldsymbol{x})$（$f(\boldsymbol{x})$的定义在式(8.1)上方一段话），因此上式应该写为

$$\begin{aligned}\ell_{\exp}(H \mid \mathcal{D}) =& \sum_{i=1}^{|D|} \mathcal{D}(\boldsymbol{x}_i) \mathbb{E}[e^{-f(\boldsymbol{x}_i)H(\boldsymbol{x}_i)}] \\ =& \sum_{i=1}^{|D|} \mathcal{D}(\boldsymbol{x}_i) \left(e^{-H(\boldsymbol{x}_i)}P(f(\boldsymbol{x}_i)=1 \mid \boldsymbol{x}_i) + e^{H(\boldsymbol{x}_i)}P(f(\boldsymbol{x}_i)=-1 \mid \boldsymbol{x}_i)\right)\end{aligned}$$

现对$H(\boldsymbol{x}_i)$求导，求和项中只有第i项$\mathcal{D}(\boldsymbol{x}_i)\mathbb{E}[e^{-f(\boldsymbol{x}_i)H(\boldsymbol{x}_i)}]$会起作用。$\mathcal{D}(\boldsymbol{x}_i)$为常量，不影响接下来式(8.7)中令导数等于零求解的过程，而

$$\frac{\partial e^{-H(\boldsymbol{x})}}{\partial H(\boldsymbol{x})} = -e^{-H(\boldsymbol{x})}, \quad \frac{\partial e^{H(\boldsymbol{x})}}{\partial H(\boldsymbol{x})} = e^{H(\boldsymbol{x})}$$

因此求导结果为

$$\frac{\partial \ell_{\exp}(H \mid \mathcal{D})}{\partial H(\boldsymbol{x})} = -e^{-H(\boldsymbol{x})}P(f(\boldsymbol{x})=1 \mid \boldsymbol{x}) + e^{H(\boldsymbol{x})}P(f(\boldsymbol{x})=-1 \mid \boldsymbol{x})$$

即式(8.6)。

4、式(8.7)的推导

令式(8.6)等于零：　$-e^{-H(\boldsymbol{x})}P(f(\boldsymbol{x})=1 \mid \boldsymbol{x}) + e^{H(\boldsymbol{x})}P(f(\boldsymbol{x})=-1 \mid \boldsymbol{x}) = 0$

移项：　$e^{H(\boldsymbol{x})}P(f(\boldsymbol{x})=-1 \mid \boldsymbol{x}) = e^{-H(\boldsymbol{x})}P(f(\boldsymbol{x})=1 \mid \boldsymbol{x})$

两边同乘$\frac{e^{H(\boldsymbol{x})}}{P(f(\boldsymbol{x})=-1|\boldsymbol{x})}$：　$e^{2H(\boldsymbol{x})} = \frac{P(f(\boldsymbol{x})=1|\boldsymbol{x})}{P(f(\boldsymbol{x})=-1|\boldsymbol{x})}$

取$\ln(\cdot)$：　$2H(\boldsymbol{x}) = \ln \frac{P(f(\boldsymbol{x})=1|\boldsymbol{x})}{P(f(\boldsymbol{x})=-1|\boldsymbol{x})}$

两边同除$\frac{1}{2}$即得式(8.7)。

式(8.6)和式(8.7)的推导在原始文献[Friedman J H, Hastie T, Tibshirani R, et al. Additive logistic regression : A statistical view of boosting[J]. Annals of Statistics, 2000, 28(2): 337-407.]第 9 页(第 345 页)也有推导：

LEMMA 1. *$E(e^{-yF(x)})$ is minimized at*

$$F(x)=\frac{1}{2}\log\frac{P(y=1|x)}{P(y=-1|x)}. \tag{12}$$

Hence

$$P(y=1|x)=\frac{e^{F(x)}}{e^{-F(x)}+e^{F(x)}}, \tag{13}$$

$$P(y=-1|x)=\frac{e^{-F(x)}}{e^{-F(x)}+e^{F(x)}}. \tag{14}$$

PROOF. While E entails expectation over the joint distribution of y and x, it is sufficient to minimize the criterion conditional on x:

$$E\left(e^{-yF(x)}|x\right)=P(y=1|x)e^{-F(x)}+P(y=-1|x)e^{F(x)},$$

$$\frac{\partial E\left(e^{-yF(x)}|x\right)}{\partial F(x)}=-P(y=1|x)e^{-F(x)}+P(y=-1|x)e^{F(x)}.$$

The result follows by setting the derivative to zero. □

5、式(8.8)的推导

该式通俗易懂，这里仅解释一下贝叶斯错误率的概念。这来源于 P148 的式(7.6)表示的贝叶斯最优分类器，可以发现式(8.8)的最终结果是式(7.6)的二分类特殊形式。

到此为止，本节证明了指数损失函数是分类任务原本 0/1 损失函数的一致的替代损失函数（参见 P131 图 6.5），而指数损失函数有更好的数学性质，例如它是连续可微函数，因此接下来的式(8.9)至式(8.19)基于指数损失函数推导 AdaBoost 的理论细节。

6、式(8.9)的推导

【注 1】本节符号略有混乱，如前所述式(8.4)的$H(\boldsymbol{x})$是连续实值函数，但在图 8.3 最后一行的输出$H(\boldsymbol{x})$明显只能取-1和$+1$两个值（与式(8.2)相同），本节除了图 8.3 最后一行的输出之外，$H(\boldsymbol{x})$均以式(8.4)的连续实值函数为准。

【注 2】乍一看本式有些问题，为什么要最小化$\ell_{\exp}(\alpha_t h_t \mid \mathcal{D}_t)$？图 8.3 中的第 3 行的表达式$h_t=\mathfrak{L}(D,\mathcal{D}_t)$不是代表着应该最小化$\ell_{\exp}(h_t \mid \mathcal{D}_t)$么？或者从整体来看，第$t$轮迭代也应该最小化$\ell_{\exp}(H_t \mid \mathcal{D})=\ell_{\exp}(H_{t-1}+\alpha_t h_t \mid \mathcal{D})$，这样最终$T$轮迭代结束后得到的式(8.4)就可以最小化$\ell_{\exp}(H \mid \mathcal{D})$了。实际上，理解了 AdaBoost 之后就会发现，$\ell_{\exp}(\alpha_t h_t \mid \mathcal{D}_t)$与$\ell_{\exp}(H_t \mid \mathcal{D})$是等价的，详见后面的“AdaBoost 的个人推导”。另外，$h_t=\mathfrak{L}(D,\mathcal{D}_t)$也是推导的结论之一，即后面的式(8.18)，而不是无缘无故靠直觉用$\mathfrak{L}(D,\mathcal{D}_t)$得到$h_t$。

暂时不管以上疑问，权且按作者思路推导一下：

第 1 个等号与式(8.5)的区别仅在于到底针对$\alpha_t h_t(\boldsymbol{x})$还是$H(\boldsymbol{x})$，代入即可；

第 2 个等号是考虑到$h_t(\boldsymbol{x})$和$f(\boldsymbol{x})$均只能取-1和$+1$两个值，其中$\mathbb{I}(\cdot)$为指示函数；

第 3 个等号对中括号的两项分别求$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}[\cdot]$，而$e^{\alpha_t}$和$e^{-\alpha_t}$与$\boldsymbol{x}$无关，可以作为常数项拿到$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}[\cdot]$外面，而$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}[\mathbb{I}(f(\boldsymbol{x})=h_t(\boldsymbol{x}))]$表示在数据集$D$上、样本权值分布为$\mathcal{D}_t$时$f(\boldsymbol{x})$和$h_t(\boldsymbol{x})$相等次数的期望，即$P_{\boldsymbol{x}\sim\mathcal{D}_t}(f(\boldsymbol{x})=h_t(\boldsymbol{x}))$，也就是正确率，即$(1-\epsilon_t)$；同理，$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}[\mathbb{I}(f(\boldsymbol{x})\neq h_t(\boldsymbol{x}))]$表示在数据集$D$上、样本权值分布为$\mathcal{D}_t$时$f(\boldsymbol{x})$和$h_t(\boldsymbol{x})$不相等次数的期望，即$P_{\boldsymbol{x}\sim\mathcal{D}_t}(f(\boldsymbol{x})\neq h_t(\boldsymbol{x}))$，也就是错误率$\epsilon_t$；

第 4 个等号即为将$P_{\boldsymbol{x}\sim\mathcal{D}_t}(f(\boldsymbol{x})=h_t(\boldsymbol{x}))$替换为$(1-\epsilon_t)$、将$P_{\boldsymbol{x}\sim\mathcal{D}_t}(f(\boldsymbol{x})\neq h_t(\boldsymbol{x}))$替换为$\epsilon_t$的结果。

7、式(8.12)的推导[?]

【注】本式确实有问题，虽然并不影响后面式(8.18)和式(8.19)的结果。AdaBoost 第t轮迭代应该求解如下优化问题从而得到α_t和$h_t(\boldsymbol{x})$:

$$(\alpha_t, h_t(\boldsymbol{x})) = \arg\min_{\alpha,h} \ell_{\exp}(H_{t-1}+\alpha h \mid \mathcal{D})$$

对于该问题，先对于固定的任意$\alpha>0$，求解$h_t(\boldsymbol{x})$；得到$h_t(\boldsymbol{x})$后再求α_t。

原始文献[Friedman J H, Hastie T, Tibshirani R, et al. Additive logistic regression : A statistical view of boosting[J]. Annals of Statistics, 2000, 28(2): 337-407.]第 10 页(第 346 页)对西瓜书中式(8.12)到式(8.12)的相关推导如下：

RESULT 1. *The Discrete AdaBoost algorithm (population version) builds an additive logistic regression model via Newton-like updates for minimizing* $E(e^{-yF(x)})$.

PROOF. Let $J(F)=E[e^{-yF(x)}]$. Suppose we have a current estimate $F(x)$ and seek an improved estimate $F(x)+cf(x)$. For fixed c (and x), we expand $J(F(x)+cf(x))$ to second order about $f(x)=0$,

$$\begin{aligned} J(F+cf) &= E[e^{-y(F(x)+cf(x))}] \\ &\approx E[e^{-yF(x)}(1-ycf(x)+c^2y^2f(x)^2/2)] \\ &= E[e^{-yF(x)}(1-ycf(x)+c^2/2)], \end{aligned}$$

since $y^2=1$ and $f(x)^2=1$. Minimizing pointwise with respect to $f(x)\in\{-1,1\}$, we write

$$f(x)=\arg\min_f E_w(1-ycf(x)+c^2/2|x). \tag{16}$$

可以发现原文献中保留了参数c（即α）。当然，对于任意$\alpha>0$，并不影响推导结果。

且不管以上问题，权且按作者思路推导一下：

本式较为简单，只要将$H_t(\boldsymbol{x})=H_{t-1}(\boldsymbol{x})+h_t(\boldsymbol{x})$替换式(8.5)中的$H(\boldsymbol{x})$即可。

8、式(8.13)的推导

复习一下泰勒展开式：

$$f(x)=\frac{f(x_0)}{0!}+\frac{f'(x_0)}{1!}(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\ldots+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+\ldots$$

当$f(x)=e^{-x}$时，在$x_0=0$处泰勒展开：

$$e^{-x}=1-x+\frac{1}{2}x^2+\ldots$$

将x换为$f(\boldsymbol{x})h_t(\boldsymbol{x})$，对$e^{-f(\boldsymbol{x})h_t(\boldsymbol{x})}$泰勒展开，代入即得式(8.13)。

实际上，此处保留一阶泰勒展开项即可，后面提到的 Gradient Boosting 理论框架就是只使用了一阶泰勒展开；当然二阶项为常数，也并不影响推导结果，原文献中也保留了二阶项。

9、式(8.14)的推导[?]

本式基于式(8.12)和式(8.13)的结果。且不管以上问题，权且按作者思路推导一下：

第 2 个等号就是将式(8.13)代入；

第 3 个等号是因为

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1-f(\boldsymbol{x})h(\boldsymbol{x})+\frac{1}{2}\right)\right]\\=&\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{3}{2}e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}-e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}f(\boldsymbol{x})h(\boldsymbol{x})\right]\\=&\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{3}{2}e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]-\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}f(\boldsymbol{x})h(\boldsymbol{x})\right]\end{aligned}$$

本式自变量为$h(\boldsymbol{x})$，而$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{3}{2}e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]$与$h(\boldsymbol{x})$无关，也就是一个常数；只需最大化第二项$-\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}f(\boldsymbol{x})h(\boldsymbol{x})\right]$即可，将负号去掉，原最小化问题变为最大化问题；

第 4 个等号仍是因为$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]$是与自变量$h(\boldsymbol{x})$无关的正常数（因为指数函数$e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}$大于零，从而求期望$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\cdot]$的结果也大于零），而最大化问题乘以某个正常数与原问题等价（例如$\arg\max_x\ (1-x^2)$与$\arg\max_x\ 2(1-x^2)$的结果均为$x=0$）；

10、式(8.16)的推导

类似于式(8.6)的推导中：

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H(\boldsymbol{x})}]=\sum_{i=1}^{|D|}\mathcal{D}(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)H(\boldsymbol{x}_i)}$$

并结合式(8.15)的定义：

$$\mathcal{D}_t(\boldsymbol{x}_i)=\mathcal{D}(\boldsymbol{x}_i)\frac{e^{-f(\boldsymbol{x}_i)H_{t-1}(\boldsymbol{x}_i)}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}$$

其中$\boldsymbol{x}_i$表示数据集D的第i个样本，故与式(8.15)形式一样。针对式(8.14)最后一行的表达式：

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}f(\boldsymbol{x})h(\boldsymbol{x})\right]\\=&\sum_{i=1}^{|D|}\mathcal{D}(\boldsymbol{x}_i)\frac{e^{-f(\boldsymbol{x}_i)H_{t-1}(\boldsymbol{x}_i)}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}f(\boldsymbol{x}_i)h(\boldsymbol{x}_i)\\=&\sum_{i=1}^{|D|}\mathcal{D}_t(\boldsymbol{x}_i)f(\boldsymbol{x}_i)h(\boldsymbol{x}_i)\\=&\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[f(\boldsymbol{x})h(\boldsymbol{x})\right]\end{aligned}$$

11、式(8.17)的推导

当$f(\boldsymbol{x})=h(\boldsymbol{x})$时，$f(\boldsymbol{x})h(\boldsymbol{x})=1$；

当$f(\boldsymbol{x})\neq h(\boldsymbol{x})$时，$f(\boldsymbol{x})h(\boldsymbol{x})=-1$；

当$f(\boldsymbol{x})=h(\boldsymbol{x})$时，$\mathbb{I}\left(f(\boldsymbol{x})\neq h(\boldsymbol{x})\right)=0$，则$1-2\mathbb{I}\left(f(\boldsymbol{x})\neq h(\boldsymbol{x})\right)=1$；

当$f(\boldsymbol{x})\neq h(\boldsymbol{x})$时，$\mathbb{I}\left(f(\boldsymbol{x})\neq h(\boldsymbol{x})\right)=1$，则$1-2\mathbb{I}\left(f(\boldsymbol{x})\neq h(\boldsymbol{x})\right)=-1$；

综上所述，等式两边相等。

12、式(8.18)的推导

本式基于式(8.17)的恒等关系，由式(8.16)推导而来。

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[f(\boldsymbol{x})h(\boldsymbol{x})\right] =&\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[1-2\mathbb{I}\left(f(\boldsymbol{x})\neq h(\boldsymbol{x})\right)\right]\\ =&\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[1\right]-2\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[\mathbb{I}\left(f(\boldsymbol{x})\neq h(\boldsymbol{x})\right)\right]\\ =&1-2\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[\mathbb{I}\left(f(\boldsymbol{x})\neq h(\boldsymbol{x})\right)\right]\end{aligned}$$

类似于式(8.14)的第 3 个和第 4 个等号，由式(8.16)的结果开始推导：

$$\begin{aligned}h_t(\boldsymbol{x}) &= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[f(\boldsymbol{x})h(\boldsymbol{x})\right]\\ &= \arg\max_h \left(1-2\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[\mathbb{I}\left(f(\boldsymbol{x})\neq h(\boldsymbol{x})\right)\right]\right)\\ &= \arg\max_h \left(-2\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[\mathbb{I}\left(f(\boldsymbol{x})\neq h(\boldsymbol{x})\right)\right]\right)\\ &= \arg\min_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[\mathbb{I}\left(f(\boldsymbol{x})\neq h(\boldsymbol{x})\right)\right]\end{aligned}$$

此式表示理想的$h_t(\boldsymbol{x})$在分布$\mathcal{D}_t$下最小化分类误差，因此有图 8.3 第 3 行$h_t(\boldsymbol{x})=\mathfrak{L}(D,\mathcal{D}_t)$，即分类器$h_t(\boldsymbol{x})$可以基于分布$\mathcal{D}_t$从数据集$D$中训练而得，而我们在训练分类器时，一般来说最小化的损失函数就是分类误差。

13、式(8.19)的推导

第 1 个等号是将式(8.15)中的t换为$t+1$（同时$t-1$换为t）；

第 2 个等号是将$H_t(\boldsymbol{x})=H_{t-1}(\boldsymbol{x})+\alpha_t h_t(\boldsymbol{x})$代入分子即可；

第 3 个等号是乘以$\frac{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}$后，凑出式(8.15)的$\mathcal{D}_t(\boldsymbol{x})$表达式，以符号$\mathcal{D}_t(\boldsymbol{x})$替换即得。到此之后，得到$\mathcal{D}_{t+1}(\boldsymbol{x})$与$\mathcal{D}_t(\boldsymbol{x})$的关系，但为了确保$\mathcal{D}_{t+1}(\boldsymbol{x})$是一个分布，需要对得到的$\mathcal{D}_{t+1}(\boldsymbol{x})$进行规范化，即图 8.3 第 7 行的$Z_t$。式(8.19)第 3 行最后一个分式将在规范化过程被吸收。

14、AdaBoost 的个人推导

以上推导感觉有点乱，个人综合各种资料，个人版 AdaBoost 推导如下：

AdaBoost 的目标是学得T个$h_t(\boldsymbol{x})$和相应的T个α_t，得到式(8.4)的$H(\boldsymbol{x})$，使式(8.5)指数损失函数$\ell_{\exp}(H\mid\mathcal{D})$最小，这就是求解所谓的“**加性模型**”。特别强调一下，分类器$h_t(\boldsymbol{x})$如何得到及其相应的权重α_t等于多少都是需要求解的（$h_t(\boldsymbol{x})=\mathfrak{L}(D,\mathcal{D}_t)$，即基于分布$\mathcal{D}_t$从数据集$D$中经过最小化训练误差训练出分类器$h_t$，也就是式(8.18)，$\alpha_t$参见式(8.18)）。

“通常这是一个复杂的优化问题（同时学得T个$h_t(\boldsymbol{x})$和相应的T个α_t很困难）。**前向分步算法**求解这一优化问题的想法是：因为学习的是加法模型，如果能够从前向后，每一步只学习一个基函数$h_t(\boldsymbol{x})$及其系数α_t，逐步逼近最小化指数损失函数$\ell_{\exp}(H\mid\mathcal{D})$，那么就可以简化优化的复杂度。”（摘自李航《统计学习方法》第 144 页，略有改动）

因此，AdaBoost 每轮迭代只需要得到一个基分类器和其投票权重，设第t轮迭代需得到基分类器$h_t(\boldsymbol{x})$，对应的投票权重为α_t，则集成分类器$H_t(\boldsymbol{x})=H_{t-1}(\boldsymbol{x})+\alpha_t h_t(\boldsymbol{x})$，其中$H_0(\boldsymbol{x})=0$。为表达式简洁，常常将$h_t(\boldsymbol{x})$简写为$h_t$，$H_t(\boldsymbol{x})$简写为$H_t$。则第$t$轮实际为如下优化问题（本节式(8.4)到式(8.8)已经证明了指数损失函数是分类任务原本 0/1 损失函数的一致替代损失函数）：

$$(\alpha_t,h_t)=\arg\min_{\alpha,h}\ell_{\exp}(H_{t-1}+\alpha h\mid\mathcal{D})$$

表示每轮得到的基分类器$h_t(\boldsymbol{x})$和对应的权重α_t是最小化集成分类器$H_t=H_{t-1}+\alpha_t h_t$在数据集D上、样本权值分布为$\mathcal{D}$（即初始化样本权值分布，也就是$\mathcal{D}_1$）时的指数损失函数

$\ell_{\exp}(H_{t-1}+\alpha h \mid \mathcal{D})$的结果。这就是**前向分步算法**求解**加性模型**的思路。

根据式(8.5)将指数损失函数表达式代入，则

$$\begin{aligned}\ell_{\exp}(H_{t-1}+\alpha h \mid \mathcal{D}) &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})(H_{t-1}(\boldsymbol{x})+\alpha h(\boldsymbol{x}))}\right] \\ &= \sum_{i=1}^{|D|}\mathcal{D}(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)(H_{t-1}(\boldsymbol{x}_i)+\alpha h(\boldsymbol{x}_i))} \\ &= \sum_{i=1}^{|D|}\mathcal{D}(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)H_{t-1}(\boldsymbol{x}_i)}e^{-f(\boldsymbol{x}_i)\alpha h(\boldsymbol{x}_i)} \\ &= \sum_{i=1}^{|D|}\mathcal{D}(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)H_{t-1}(\boldsymbol{x}_i)}\left(e^{-\alpha}\mathbb{I}(f(\boldsymbol{x}_i)=h(\boldsymbol{x}_i))+e^{\alpha}\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))\right)\end{aligned}$$

上式推导中，由于$f(\boldsymbol{x}_i)$和$h(\boldsymbol{x}_i)$均只能取$-1,+1$两个值，因此当$f(\boldsymbol{x}_i)=h(\boldsymbol{x}_i)$时，$f(\boldsymbol{x}_i)h(\boldsymbol{x}_i)=1$，当$f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i)$时，$f(\boldsymbol{x}_i)h(\boldsymbol{x}_i)=-1$。另外，$f(\boldsymbol{x}_i)$和$h(\boldsymbol{x}_i)$要么相等，要么不相等，二者只能有一个为真， 因此以下等式恒成立：

$$\mathbb{I}(f(\boldsymbol{x}_i)=h(\boldsymbol{x}_i))+\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))=1$$

所以

$$\begin{aligned}&e^{-\alpha}\mathbb{I}(f(\boldsymbol{x}_i)=h(\boldsymbol{x}_i))+e^{\alpha}\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i)) \\ =&e^{-\alpha}\mathbb{I}(f(\boldsymbol{x}_i)=h(\boldsymbol{x}_i))+e^{-\alpha}\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))-e^{-\alpha}\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))+e^{\alpha}\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i)) \\ =&e^{-\alpha}\left(\mathbb{I}(f(\boldsymbol{x}_i)=h(\boldsymbol{x}_i))+\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))\right)+(e^{\alpha}-e^{-\alpha})\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i)) \\ =&e^{-\alpha}+(e^{\alpha}-e^{-\alpha})\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))\end{aligned}$$

将此结果代入$\ell_{\exp}(H_{t-1}+\alpha h \mid \mathcal{D})$，得（注：以下表达式后面求解权重$\alpha_l$时仍会使用）

$$\begin{aligned}\ell_{\exp}(H_{t-1}+\alpha h \mid \mathcal{D}) &= \sum_{i=1}^{|D|}\mathcal{D}(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)H_{t-1}(\boldsymbol{x}_i)}\left(e^{-\alpha}+(e^{\alpha}-e^{-\alpha})\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))\right) \\ &= \sum_{i=1}^{|D|}\mathcal{D}(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)H_{t-1}(\boldsymbol{x}_i)}e^{-\alpha}+\sum_{i=1}^{|D|}\mathcal{D}(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)H_{t-1}(\boldsymbol{x}_i)}(e^{\alpha}-e^{-\alpha})\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i)) \\ &= e^{-\alpha}\sum_{i=1}^{|D|}\mathcal{D}'_t(\boldsymbol{x}_i)+(e^{\alpha}-e^{-\alpha})\sum_{i=1}^{|D|}\mathcal{D}'_t(\boldsymbol{x}_i)\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))\end{aligned}$$

上式中，$\mathcal{D}'_t(\boldsymbol{x}_i)=\mathcal{D}(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)H_{t-1}(\boldsymbol{x}_i)}$，$e^{\alpha}$和$e^{-\alpha}$与求和变量$i$无关，故可以拿到求和号外面；第一项$e^{-\alpha}\sum_{i=1}^{|D|}\mathcal{D}'_t(\boldsymbol{x}_i)$与$h(\boldsymbol{x})$无关，因此对于任意$\alpha>0$，使$\ell_{\exp}(H_{t-1}+\alpha h \mid \mathcal{D})$最小的$h(\boldsymbol{x})$只需要使第二项最小即可，即

$$h_t=\arg\min_h\ (e^{\alpha}-e^{-\alpha})\sum_{i=1}^{|D|}\mathcal{D}'_t(\boldsymbol{x}_i)\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))$$

对于任意$\alpha>0$，有$e^{\alpha}-e^{-\alpha}>0$，所以上式中与$h(\boldsymbol{x})$无关的正系数可以省略：

$$h_t=\arg\min_h\ \sum_{i=1}^{|D|}\mathcal{D}'_t(\boldsymbol{x}_i)\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))$$

此即式(8.18)另一种表达形式。注意，为了确保$\mathcal{D}'_t(\boldsymbol{x})$是一个分布，需要对其进行规范化，即$\mathcal{D}_t(\boldsymbol{x})=\frac{\mathcal{D}'_t(\boldsymbol{x})}{Z_t}$，然而规范化因子$Z_t=\sum_{i=1}^{|D|}\mathcal{D}'_t(\boldsymbol{x}_i)$为常数，并不影响最小化的求解。正是基于此结论，AdaBoost 通过$h_t=\mathfrak{L}(D,\mathcal{D}_t)$得到第$t$轮的基分类器（图 8.3 的第 3 行）。

由$\mathcal{D}_t(\boldsymbol{x}_i)=\mathcal{D}(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)H_{t-1}(\boldsymbol{x}_i)}$可知（规范化只是除以常数项）

$$\begin{aligned}\mathcal{D}_{t+1}(\boldsymbol{x}_i) =& \mathcal{D}(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)H_t(\boldsymbol{x}_i)}\\ =& \mathcal{D}(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)(H_{t-1}(\boldsymbol{x}_i)+\alpha_t h_t(\boldsymbol{x}_i))}\\ =& \mathcal{D}(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)H_{t-1}(\boldsymbol{x}_i)}e^{-f(\boldsymbol{x}_i)\alpha_t h_t(\boldsymbol{x}_i)}\\ =& \mathcal{D}_t(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)\alpha_t h_t(\boldsymbol{x}_i)}\end{aligned}$$

此即类似式(8.19)的分布权重更新公式。

现在只差权重α_t表达式待求。对指数损失函数$\ell_{\exp}(H_{t-1}+\alpha h_t \mid \mathcal{D})$求导，得

$$\begin{aligned}\frac{\partial \ell_{\exp}(H_{t-1}+\alpha h_t \mid \mathcal{D})}{\partial \alpha} &= \frac{\partial \left(e^{-\alpha}\sum_{i=1}^{|D|}\mathcal{D}'_t(\boldsymbol{x}_i)+(e^{\alpha}-e^{-\alpha})\sum_{i=1}^{|D|}\mathcal{D}'_t(\boldsymbol{x}_i)\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))\right)}{\partial \alpha}\\ &= -e^{-\alpha}\sum_{i=1}^{|D|}\mathcal{D}'_t(\boldsymbol{x}_i)+(e^{\alpha}+e^{-\alpha})\sum_{i=1}^{|D|}\mathcal{D}'_t(\boldsymbol{x}_i)\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))\end{aligned}$$

令导数等于零，得

$$\begin{aligned}\frac{e^{-\alpha}}{e^{\alpha}+e^{-\alpha}} =& \frac{\sum_{i=1}^{|D|}\mathcal{D}'_t(\boldsymbol{x}_i)\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))}{\sum_{i=1}^{|D|}\mathcal{D}'_t(\boldsymbol{x}_i)} = \sum_{i=1}^{|D|}\frac{\mathcal{D}'_t(\boldsymbol{x}_i)}{Z_t}\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))\\ =& \sum_{i=1}^{|D|}\mathcal{D}_t(\boldsymbol{x}_i)\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i)) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))\right]\\ =& \epsilon_t\end{aligned}$$

对上述等式化简，得

$$\begin{aligned}\frac{e^{-\alpha}}{e^{\alpha}+e^{-\alpha}} = \frac{1}{e^{2\alpha}+1} \Rightarrow e^{2\alpha}+1 = \frac{1}{\epsilon_t} \Rightarrow e^{2\alpha} = \frac{1-\epsilon_t}{\epsilon_t} \Rightarrow 2\alpha = \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)\\ \Rightarrow \alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)\end{aligned}$$

即式(8.11)。从该式可以发现，当$\epsilon_t = 1$时，$\alpha_t \to \infty$，此时集成分类器将由基分类器h_t决定，而这很可能是由于过拟合产生的结果，例如不剪枝决策树，如果一直分下去，一般情况下总能得到在训练集上分类误差很小甚至为 0 的分类器，但这并没有什么意义。所以一般在 AdaBoost 中使用弱分类器，如决策树桩（即单层决策树）。

另外，由以上指数损失函数$\ell_{\exp}(H_{t-1}+\alpha h \mid \mathcal{D})$的推导可以发现

$$\begin{aligned}\ell_{\exp}(H_{t-1}+\alpha h \mid \mathcal{D}) &= \sum_{i=1}^{|D|}\mathcal{D}(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)H_{t-1}(\boldsymbol{x}_i)}e^{-f(\boldsymbol{x}_i)\alpha h(\boldsymbol{x}_i)}\\ &= \sum_{i=1}^{|D|}\mathcal{D}'_t(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)\alpha h(\boldsymbol{x}_i)}\end{aligned}$$

这与指数损失函数$\ell_{\exp}(\alpha_t h_t \mid \mathcal{D}_t)$的表达式基本一致：

$$\begin{aligned}\ell_{\exp}(\alpha_t h_t \mid \mathcal{D}_t) &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})}\right]\\ &= \sum_{i=1}^{|D|}\mathcal{D}_t(\boldsymbol{x}_i)e^{-f(\boldsymbol{x}_i)\alpha_t h_t(\boldsymbol{x}_t)}\end{aligned}$$

而$\mathcal{D}'_t(\boldsymbol{x})$的规范化过程并不影响对$\ell_{\exp}(H_{t-1}+\alpha h \mid \mathcal{D})$求最小化操作，因此最小化式(8.9)等价于最小化$\ell_{\exp}(H_{t-1}+\alpha h \mid \mathcal{D})$，这就是式(8.9)的来历，故并无问题。

到此为止，就逐一完成了图 8.3 中第 3 行的h_t的训练（并计算训练误差）、第 6 行的权重α_t计算公式以及第 7 行的分布$\mathcal{D}_t$更新公式来历的理论推导。

15、更深一步理解权重更新公式的含义

文献[Friedman J H, Hastie T, Tibshirani R, et al. Additive logistic regression : A statistical view of boosting[J]. Annals of Statistics, 2000, 28(2): 337-407.]第 12 页(第 348 页)有如下推论：

COROLLARY 2. *After each update to the weights, the* weighted *misclassification error of the most recent weak learner is* 50%.

PROOF. This follows by noting that the c that minimizes $J(F+cf)$ satisfies

$$\frac{\partial J(F+cf)}{\partial c} = -E[e^{-y(F(x)+cf(x))}yf(x)] = 0. \tag{21}$$

The result follows since $yf(x)$ is 1 for a correct and -1 for an incorrect classification. □

即$P_{\boldsymbol{x}\sim\mathcal{D}_t}(h_{t-1}(\boldsymbol{x}) \neq f(\boldsymbol{x})) = 0.5$。用通俗的话来说就是，$h_{t-1}$**在数据集$D$上、分布为$\mathcal{D}_t$时的分类误差为 0.5**，即相当于随机猜测（最糟糕的二分类器是分类误差为 0.5，当二分类器分类误差为 1 时相当于分类误差为 0，因为将预测结果反过来用就是了）。而h_t由式(8.18)得到

$$h_t = \arg\min_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[\mathbb{I}\left(f(\boldsymbol{x}) \neq h(\boldsymbol{x})\right)\right] = \arg\min_h P_{\boldsymbol{x}\sim\mathcal{D}_t}(h(\boldsymbol{x}) \neq f(\boldsymbol{x}))$$

即h_t**是在数据集D上、分布为$\mathcal{D}_t$时分类误差最小的分类器**，因此在数据集D上、分布为$\mathcal{D}_t$时，h_t是最好的分类器，而h_{t-1}是最差的分类器，故二者差别最大。第 8.1 节的图 8.2 形象的说明了“集成个体应‘好而不同’”，此时可以说h_{t-1}和h_t非常“不同”。证明如下：

对于h_{t-1}来说，分类误差ϵ_{t-1}为

$$\begin{aligned}\epsilon_{t-1} =& P_{\boldsymbol{x}\sim\mathcal{D}_{t-1}}(h_{t-1}(\boldsymbol{x}) \neq f(\boldsymbol{x})) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{t-1}}\left[\mathbb{I}\left(h_{t-1}(\boldsymbol{x}) \neq f(\boldsymbol{x})\right)\right] \\ =& \sum_{i=1}^{|D|}\mathcal{D}_{t-1}(\boldsymbol{x}_i)\mathbb{I}\left(h_{t-1}(\boldsymbol{x}) \neq f(\boldsymbol{x})\right) \\ =& \frac{\sum_{i=1}^{|D|}\mathcal{D}_{t-1}(\boldsymbol{x}_i)\mathbb{I}\left(h_{t-1}(\boldsymbol{x}) \neq f(\boldsymbol{x})\right)}{\sum_{i=1}^{|D|}\mathcal{D}_{t-1}(\boldsymbol{x}_i)\mathbb{I}\left(h_{t-1}(\boldsymbol{x}) = f(\boldsymbol{x})\right) + \sum_{i=1}^{|D|}\mathcal{D}_{t-1}(\boldsymbol{x}_i)\mathbb{I}\left(h_{t-1}(\boldsymbol{x}) \neq f(\boldsymbol{x})\right)}\end{aligned}$$

在第t轮，根据分布更新公式(8.19)或图 3 第 7 行（规范化因子Z_{t-1}为常量）：

$$\mathcal{D}_t = \frac{\mathcal{D}_{t-1}}{Z_{t-1}}e^{-f(\boldsymbol{x})\alpha_{t-1}h_{t-1}(\boldsymbol{x})}$$

其中根据式(8.11)，第$t-1$轮的权重

$$\alpha_{t-1} = \frac{1}{2}\ln\frac{1-\epsilon_{t-1}}{\epsilon_{t-1}} = \ln\sqrt{\frac{1-\epsilon_{t-1}}{\epsilon_{t-1}}}$$

代入$\mathcal{D}_t$的表达式，则

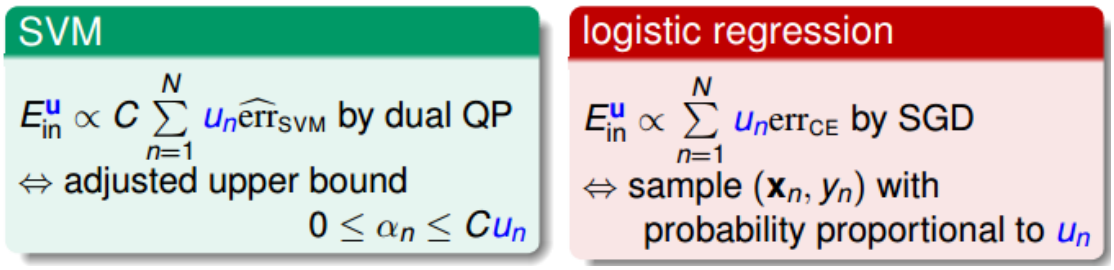
$$\mathcal{D}_t = \begin{cases}\frac{\mathcal{D}_{t-1}}{Z_{t-1}}\cdot\sqrt{\frac{\epsilon_{t-1}}{1-\epsilon_{t-1}}} & ,\text{if } h_{t-1}(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \frac{\mathcal{D}_{t-1}}{Z_{t-1}}\cdot\sqrt{\frac{1-\epsilon_{t-1}}{\epsilon_{t-1}}} & ,\text{if } h_{t-1}(\boldsymbol{x}) \neq f(\boldsymbol{x})\end{cases}$$

那么h_{t-1}在数据集D上、分布为$\mathcal{D}_t$时的分类误差$P_{\boldsymbol{x}\sim\mathcal{D}_t}(h_{t-1}(\boldsymbol{x}) \neq f(\boldsymbol{x}))$为（注意，下式第二行的分母等于 1，因为$\mathbb{I}\left(h_{t-1}(\boldsymbol{x}) = f(\boldsymbol{x})\right) + \mathbb{I}\left(h_{t-1}(\boldsymbol{x}) \neq f(\boldsymbol{x})\right) = 1$）

$$
\begin{aligned}
&P_{\boldsymbol{x}\sim\mathcal{D}_t}(h_{t-1}(\boldsymbol{x})\neq f(\boldsymbol{x}))=\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[\mathbb{I}\left(h_{t-1}(\boldsymbol{x})\neq f(\boldsymbol{x})\right)\right]\\
=&\frac{\sum_{i=1}^{|D|}\mathcal{D}_t(\boldsymbol{x}_i)\mathbb{I}\left(h_{t-1}(\boldsymbol{x}_i)\neq f(\boldsymbol{x}_i)\right)}{\sum_{i=1}^{|D|}\mathcal{D}_t(\boldsymbol{x}_i)\mathbb{I}\left(h_{t-1}(\boldsymbol{x}_i)=f(\boldsymbol{x}_i)\right)+\sum_{i=1}^{|D|}\mathcal{D}_t(\boldsymbol{x}_i)\mathbb{I}\left(h_{t-1}(\boldsymbol{x}_i)\neq f(\boldsymbol{x}_i)\right)}\\
=&\frac{\sum_{i=1}^{|D|}\frac{\mathcal{D}_{t-1}(\boldsymbol{x}_i)}{Z_{t-1}}\cdot\sqrt{\frac{1-\epsilon_{t-1}}{\epsilon_{t-1}}}\mathbb{I}\left(h_{t-1}(\boldsymbol{x}_i)\neq f(\boldsymbol{x}_i)\right)}{\sum_{i=1}^{|D|}\frac{\mathcal{D}_{t-1}(\boldsymbol{x}_i)}{Z_{t-1}}\cdot\sqrt{\frac{\epsilon_{t-1}}{1-\epsilon_{t-1}}}\mathbb{I}\left(h_{t-1}(\boldsymbol{x}_i)=f(\boldsymbol{x}_i)\right)+\sum_{i=1}^{|D|}\frac{\mathcal{D}_{t-1}(\boldsymbol{x}_i)}{Z_{t-1}}\cdot\sqrt{\frac{1-\epsilon_{t-1}}{\epsilon_{t-1}}}\mathbb{I}\left(h_{t-1}(\boldsymbol{x}_i)\neq f(\boldsymbol{x}_i)\right)}\\
=&\frac{\sqrt{\frac{1-\epsilon_{t-1}}{\epsilon_{t-1}}}\cdot\sum_{i=1}^{|D|}\mathcal{D}_{t-1}(\boldsymbol{x}_i)\mathbb{I}\left(h_{t-1}(\boldsymbol{x}_i)\neq f(\boldsymbol{x}_i)\right)}{\sqrt{\frac{\epsilon_{t-1}}{1-\epsilon_{t-1}}}\cdot\sum_{i=1}^{|D|}\mathcal{D}_{t-1}(\boldsymbol{x}_i)\mathbb{I}\left(h_{t-1}(\boldsymbol{x}_i)=f(\boldsymbol{x}_i)\right)+\sqrt{\frac{1-\epsilon_{t-1}}{\epsilon_{t-1}}}\cdot\sum_{i=1}^{|D|}\mathcal{D}_{t-1}(\boldsymbol{x}_i)\mathbb{I}\left(h_{t-1}(\boldsymbol{x}_i)\neq f(\boldsymbol{x}_i)\right)}\\
=&\frac{\sqrt{\frac{1-\epsilon_{t-1}}{\epsilon_{t-1}}}\cdot\epsilon_{t-1}}{\sqrt{\frac{\epsilon_{t-1}}{1-\epsilon_{t-1}}}\cdot(1-\epsilon_{t-1})+\sqrt{\frac{1-\epsilon_{t-1}}{\epsilon_{t-1}}}\cdot\epsilon_{t-1}}=\frac{1}{2}
\end{aligned}
$$

16、能够接受带权样本的基学习算法

在林轩田老师的《机器学习技法》课程第 8 讲提到了两种能够接受带权样本的基学习算法，分别是 SVM 和基于随机梯度下降(SGD)的对率回归：



其实原理很简单：对于 SVM 来说，针对 P130 页的优化目标式(6.29)来说，第二项为损失项，此时每个样本的损失$\ell_{0/1}(y_i(\boldsymbol{w}^\top\boldsymbol{x}_i+b)-1)$直接相加，即样本权值分布为$\mathcal{D}(\boldsymbol{x}_i)=\frac{1}{m}$，其中$m$为数据集$D$样本个数；若样本权值更新为$\mathcal{D}_t(\boldsymbol{x}_i)$，则此时损失求和项应该变为

$$\sum_{i=1}^{m} m\mathcal{D}_t(\boldsymbol{x}_i)\cdot\ell_{0/1}(y_i(\boldsymbol{w}^\top\boldsymbol{x}_i+b)-1)$$

若将$\mathcal{D}(\boldsymbol{x}_i)=\frac{1}{m}$替换$\mathcal{D}_t(\boldsymbol{x}_i)$，则就是每个样本的损失$\ell_{0/1}(y_i(\boldsymbol{w}^\top\boldsymbol{x}_i+b)-1)$直接相加。

如此更改后，最后推导结果影响的是式(6.39)，将由$C=\alpha_i+\mu_i$变为

$$C\cdot m\mathcal{D}_t(\boldsymbol{x}_i)=\alpha_i+\mu_i$$

进而由$\alpha_i,\mu_i\ge 0$导出$0\le\alpha_i\le C\cdot m\mathcal{D}_t(\boldsymbol{x}_i)$。上述林老师课程中的$u_n=m\mathcal{D}_t(\boldsymbol{x}_n)$。

对于基于随机梯度下降(SGD)的对率回归，每次随机选择一个样本进行梯度下降，总体上的期望损失即为式(3.27)，此时每个样本被选到的概率相同，相当于$\mathcal{D}(\boldsymbol{x}_i)=\frac{1}{m}$。若样本权值更新为$\mathcal{D}_t(\boldsymbol{x}_i)$，则类似于 SVM，针对式(3.27)只需要给第$i$项乘以$m\mathcal{D}_t(\boldsymbol{x}_i)$即可，相当于每次随机梯度下降选择样本时以概率$\mathcal{D}_t(\boldsymbol{x}_i)$选择样本$\boldsymbol{x}_i$即可。

注意，这里总的损失中出现了样本个数m。这是因为在定义损失时未求均值，若对式(6.29)的第二项和式(3.27)乘以$\frac{1}{m}$则可以将m抵消掉。然而常数项在最小化式(3.27)实际上并不影响什么，对于式(6.29)来说只要选择平衡参数C时选为原来的m倍即可。

当然，正如 P177 第三段中所说，“对无法接受带权样本的基学习算法，则可通过‘重采样法’来处理，即在每一轮学习中，根据样本分布对训练集重新进行采样，再用重采样而得的样本集对基学习器进行训练”。注意，“重采样法”想要得到数据分布为$\mathcal{D}(\boldsymbol{x})$的数据集，而非以$\mathcal{D}(\boldsymbol{x})$对原数据集进行 bootstrap 采样，二者区别详见下一节“Bagging 算法的解释”。

8.3 Bagging 与随机森林

本节内容通俗易懂，几乎不需要什么注解。

1、Bagging 算法的解释

如 8.3.1 节所述，Bagging 算法采用有放回采样方式，生成T个含m个训练样本的采样集（其中T为基分类器个数，m训练集样本个数），由第 2 章式(2.1)可知，生成的采样集包含约 63.2%的训练集样本。在 Matlab 中可用如下几行代码实现 bootstrap sampling 的过程：

```
idx_bag = zeros(m,1); %m is the number of training examples
for ii=1:m
    idx_bag(ii)=randi(m);%bootstrap sampling
end
idx_oob = setdiff(1:m,idx_bag);
```

以上得到了包内样本的索引 idx_bag，只需按此索引从训练集中得到与训练集样本数量相同的包内样本集合即可，剩余样本索引存在 idx_oob 中，可以发现约有 0.368*m 个样本；若要训练T个基分类器，只需重复以上过程T次即可。

实际使用中，个人经常采用类似 2.2.1 节的留出法，直接随机选出训练集 67%的样本（三分之二）用于训练，在 Matlab 中可采用如下命令完成（实际已不再是 bootstrap）：

```
n_rand = randperm(m);%m is the number of training examples
idx_train = n_rand(1:round(m*0.67));
idx_test = setdiff(1:m, idx_train);
```

以上得到了采样集的索引 idx_train，只需按此索引从训练集中得到采样集即可，剩余样本索引存在 idx_test 中；若要训练T个基分类器，只需重复以上过程T次即可。

以上实现简单直接，易于理解采样过程。实际上 Matlab 自带有 randsample 函数可以实现以上两种过程。对于标准 Bagging 使用的 bootstrap sampling：

```
w = ones(1,m);%weights
idx_bag = randsample(1:m,m,'true',w);
idx_oob = setdiff(1:m,idx_bag);
```

第 2 行表示从整数集合 1~m（第 1 个参数 1:m）中以对应权重 w（第 4 个参数 w），用有放回采样方式（第 3 个参数'true'）随机选取 m 个样本（第 2 个参数 m）；此处权重 w 不用规范化，Matlab 在使用时会自己除以 sum(w)从而将权重变为一个概率分布；randsample 可以实现针对每个样本以不同的权重进行采样，这一点使用起来很方便。

对于第二种以无放回采样方式选取训练集部分样本（类似留出法）：

```
idx_train = randsample(1:m,round(m*0.67));
idx_test = setdiff(1:m,idx_train);
```

第 1 行表示从整数集合 1~m（第 1 个参数 1:m）中以均匀分布，用无放回采样方式（无第 3 个参数，即默认为'false'）随机选取 round(m*0.67)个样本（第 2 个参数 round(m*0.67)）；randsample 不支持带权的无放回采样方式，即每个样本被采样的概率相等。

以上两种采样方法是有区别的，分析如下：

第 1 种标准 Bagging 使用的 bootstrap 采样使用原来约 63.2%的样本，但采样集样本总数量保持与原样本集相同，也就是说必然有些样本在采样集中重复出现多次，因此正如西瓜书第 27 页 2.2.3 节最后一行所说，“自助法产生的数据集改变了初始数据集的分布”，即若以数据分布$\mathcal{D}(\boldsymbol{x})$（即上述使用 randsample 实现时的参数 w）对数据集D进行 bootstrap 采样得到包含样本个数仍为$|D|$的采样集D'，则D'的数据分布一定不再是$\mathcal{D}(\boldsymbol{x})$。在图 8.5 中，伪代码第 2 行的$\mathcal{D}_{bs}$可简单理解为每个样本在当前采样集中出现的概率，按 Bootstrap sampling 方法，初始训练集中约有 63.2%的样本出现在采样集中，注意采样集与初始训练集样本个数相同，也就是说初始训练集中的有些样本在采样集中出现多次，有些样本在采样集中出现一次，有些样本在采样集中没有出现，初始训练集中每个样本在采样集中出现的频次除以样本个数m即组成$\mathcal{D}_{bs}$，即自助采样产生的样本分布。

第 2 种类似的留出法只使用原数据集的部分样本，且每个样本只出现一次，即在采样集中每个样本权重相同。虽然以上无论是基于 randperm 函数的实现还是 randsample 函数的实现，都是以均匀分布从原数据集中取出约 67%的样本，但即使采用一些特殊手段以非均匀分布从原数据集中取出约 67%的样本，到了采样集中每个样本仍只出现一次，若使用当前采样集直接训练学习器，仍是相当于赋予采样集中每个样本相同的权重。

最后，特别提一下在实现 AdaBoost 时，“对无法接受带权样本的基学习算法”，“通过‘重采样法’来处理”时使用的“**重采样法**”。注意图 8.3 第 3 行的$h_t = \mathfrak{L}(D, \mathcal{D}_t)$是“基于分布$\mathcal{D}_t$从数据集$D$中训练出分类器$h_t$”，也就是说“重采样法”需要从数据集$D$中得到一个数据分布为$\mathcal{D}_t$的采样集$D'$（这是“重采样法”的目标），而不是以数据分布$\mathcal{D}_t$对数据集$D$进行 bootstrap 采样或其它采样。若想实现“重采样法”的目标，可简单按如下例子方法实现：

例：数据集D包含 5 个样本$\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5$，想通过“重采样法”实现数据分布$\mathcal{D} = [0.1, 0.2, 0.3, 0.2, 0.2]$的采样集$D'$，则

$$D' = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_3, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_5\}$$

即以数据分布为比例简单扩充数据集D得到采样集D'。

另外，以上分析和代码实现只是个人看法和偏好，仅供参考，勿被误导^_^

2、式(8.20)的解释

仅注意一点即可，每个基分类器对应的 36.8%的剩余样本均是不同的（可能会有样本出现在多个基分类器对应的 36.8%的剩余样本集合中），因此总体考虑包外估计时，整体考虑的样本肯定大于训练集 36.8%的样本个数，但这些样本并不一定是所有T个基分类器的包外样本，因此在式(8.20)投票时，针对每个样本$\boldsymbol{x}$不会有T票出现（$\mathbb{I}(\boldsymbol{x} \notin D_t)$一般不会$T$次均为真），即“仅考虑那些未使用$\boldsymbol{x}$训练的基学习器在$\boldsymbol{x}$上的预测”。

3、式(8.21)的推导

本式直接使用式(8.20)的包外预测结果。但本式直接除以$|D|$（训练集D样本个数），也就是说此处假设T个基分类器的各自的包外样本的并集一定为训练集D。实际上，这个事实成立的概率也是比较大的，可以计算一下：样本属于包内的概率为 0.632，那么T次独立的随机采样均属于包内的概率为0.632^T，当$T = 5$时，$0.632^T \approx 0.1$，当$T = 10$时，$0.632^T \approx 0.01$，这么来看的话T个基分类器的各自的包外样本的并集为训练集D的概率的确挺大的。

4、随机森林的解释

在 8.3.2 节开篇第一句话就解释了随机森林的概念：随机森林是 Bagging 的一个扩展变体，是以决策树为基学习器构建 Bagging 集成的基础上，进一步在决策树的训练过程中引入

了随机属性选择。

完整版随机森林当然更复杂，这时只须知道两个重点：(1) 以决策树为基学习器；(2) 在基学习器训练过程中，选择划分属性时只使用当前结点属性集合的一个子集。

8.4 结合策略

本节内容通俗易懂，几乎不需要什么注解。

1、硬投票和软投票的解释

书中第 183 页提到了硬投票(hard voting)和软投票(soft voting)，本页左侧注释也提到多数投票法的英文术语使用不太一致，有文献称为 majority voting。本人看到有些文献中，硬投票使用 majority voting（多数投票），软投票使用 probability voting（概率投票），所以还是具体问题具体分析比较稳妥。

2、元学习器(meta-learner)的解释

书中第 183 页最后一行提到了元学习器(meta-learner)，简单解释一下，因为理解 meta 的含义有时对于理解论文中的核心思想很有帮助。

元(meta)，非常抽象，例如此处的含义，即次级学习器，或者说基于学习器结果的学习器；另外还有元语言，就是描述计算机语言的语言，还有元数学，研究数学的数学等等；

另外，论文中经常出现的还有 meta-strategy，即元策略或元方法，比如说你的研究问题是多分类问题，那么你提出了一种方法，例如对输入特征进行变换（或对输出类别做某种变换），然后再基于普通的多分类方法进行预测，这时你的方法可以看成是一种通用的框架，它虽然针对多分类问题开发，但它需要某个具体多分类方法配合才能实现，那么这样的方法是一种更高层级的方法，可以称为是一种 meta-strategy。

不太理解没关系，等看论文中碰到了再回来看看 stacking 算法，细细品味，定能悟道。

3、Stacking 算法的解释

该算法其实非常简单，对于数据集，试想你现在有了T个基分类器预测结果，也就是说数据集中的每个样本均有T个预测结果，那么怎么结合这T个预测结果呢？

本节名为“结合策略”，告诉你各种结合方法，但其实最简单的方法就是基于这T个预测结果再进行一次学习，即针对每个样本，将这T个预测结果作为输入特征，类别仍为原来的类别，既然无法抉择如何将这些结果进行结合，那么就“学习”一下吧。

图 8.9 伪代码第 9 行中$\big((z_{i1}, z_{i2}, \dots, z_{iT}), y_i\big)$将第$i$个样本进行变换，特征为$T$个基学习器的输出$(z_{i1}, z_{i2}, \dots, z_{iT})$，类别标记仍为原来的$y_i$，将所有训练集$D$中的样本进行转换得到新的数据集$D'$后，再基于$D'$进行一次学习即可，也就是 Stacking 算法。

至于说图 8.9 中伪代码第 1 行到第 3 行使用的数据集与第 5 行到第 10 行使用的数据集D之间的关系，在图 8.9 下方的一段话有详细的讨论，不再赘述。

8.5 多样性

若无特别需求，本节内容只要了解 8.5.3 节介绍的有关多样性增强的一些技巧即可。

1、式(8.28)的解释

该式本身并无太多疑问，但该式上方一行话“则集成的‘分歧’是”是令人困惑的，为

什么集成的“分歧”是式(8.28)呢？集成不就是$H(\boldsymbol{x})$么？

个人感觉此处更应该写为“则个体学习器‘分歧’的加权均值是”，此表述在式(8.36)上方解释$\overline{A}$时被使用，否则可以会有点令人费解。

2、式(8.31)与式(8.28)的等价性推导

式(8.31)与式(8.28)均为$\overline{A}(h \mid \boldsymbol{x})$，因此应该是等价的，接下来进行推导。

(1)首先，化简式(8.31)：

$$\begin{aligned}\overline{A}(h \mid \boldsymbol{x}) &= \sum_{i=1}^{T} w_i E(h_i \mid \boldsymbol{x}) - E(H \mid \boldsymbol{x}) \\ &= \sum_{i=1}^{T} w_i \left[f(\boldsymbol{x}) - h_i(\boldsymbol{x})\right]^2 - \left[f(\boldsymbol{x}) - H(\boldsymbol{x})\right]^2 \\ &= \sum_{i=1}^{T} \left[w_i (f(\boldsymbol{x}))^2 - 2 w_i f(\boldsymbol{x}) h_i(\boldsymbol{x}) + w_i (h_i(\boldsymbol{x}))^2\right] \\ &\quad - \left[(f(\boldsymbol{x}))^2 - 2 f(\boldsymbol{x}) H(\boldsymbol{x}) + (H(\boldsymbol{x}))^2\right] \\ &= (f(\boldsymbol{x}))^2 \sum_{i=1}^{T} w_i - 2 f(\boldsymbol{x}) \sum_{i=1}^{T} w_i h_i(\boldsymbol{x}) + \sum_{i=1}^{T} w_i (h_i(\boldsymbol{x}))^2 \\ &\quad - (f(\boldsymbol{x}))^2 + 2 f(\boldsymbol{x}) H(\boldsymbol{x}) - (H(\boldsymbol{x}))^2\end{aligned}$$

根据式(8.23)定义$H(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x})$，注意约束条件$\sum_{i=1}^{T} w_i = 1$，因此

$$(f(\boldsymbol{x}))^2 \sum_{i=1}^{T} w_i = (f(\boldsymbol{x}))^2, \ f(\boldsymbol{x}) \sum_{i=1}^{T} w_i h_i(\boldsymbol{x}) = f(\boldsymbol{x}) H(\boldsymbol{x})$$

代入上式化简结果，得

$$\overline{A}(h \mid \boldsymbol{x}) = \sum_{i=1}^{T} w_i (h_i(\boldsymbol{x}))^2 - (H(\boldsymbol{x}))^2$$

(2)接下来，化简式(8.28)：

$$\begin{aligned}\overline{A}(h \mid \boldsymbol{x}) &= \sum_{i=1}^{T} w_i \left[h_i(\boldsymbol{x}) - H(\boldsymbol{x})\right]^2 \\ &= \sum_{i=1}^{T} w_i (h_i(\boldsymbol{x}))^2 - \sum_{i=1}^{T} 2 w_i h_i(\boldsymbol{x}) H(\boldsymbol{x}) + \sum_{i=1}^{T} w_i (H(\boldsymbol{x}))^2 \\ &= \sum_{i=1}^{T} w_i (h_i(\boldsymbol{x}))^2 - 2 H(\boldsymbol{x}) \sum_{i=1}^{T} w_i h_i(\boldsymbol{x}) + (H(\boldsymbol{x}))^2 \sum_{i=1}^{T} w_i\end{aligned}$$

将$H(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x})$和$\sum_{i=1}^{T} w_i = 1$代入上式化简结果，得

$$\begin{aligned}\overline{A}(h \mid \boldsymbol{x}) &= \sum_{i=1}^{T} w_i (h_i(\boldsymbol{x}))^2 - 2 H(\boldsymbol{x}) H(\boldsymbol{x}) + (H(\boldsymbol{x}))^2 \\ &= \sum_{i=1}^{T} w_i (h_i(\boldsymbol{x}))^2 - (H(\boldsymbol{x}))^2\end{aligned}$$

可以发现，式(8.28)的化简结果与式(8.31)的化简结果相等，即两种定义等价。

3、式(8.32)的解释

式(8.31)的意义在于，对于示例$\boldsymbol{x}$有$\overline{A}(h \mid \boldsymbol{x}) = \overline{E}(h \mid \boldsymbol{x}) - E(H \mid \boldsymbol{x})$成立，即个体学习器分歧的加权均值等于个体学习器误差的加权均值减去集成$H(\boldsymbol{x})$的误差。

将这个结论应用于全样本上，即为式(8.32)。

例如$A_i = \int A(h_i \mid \boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}$，这是将$\boldsymbol{x}$作为连续变量来处理的，所以这里是概率密度$p(\boldsymbol{x})$和积分号；若按离散变量来处理，则变为$A_i = \sum_{\boldsymbol{x}\in D} A(h_i \mid \boldsymbol{x})p_{\boldsymbol{x}}$；其实高等数学中讲过，积分就是连续求和。

4、列联表(contingency table)的解释

第 187 页第 2 行出现了“列联表”，看论文时偶尔能遇到该术语。列联表是统计学中双变量相关分析时常见的概念。例如，现在要调查某医院各职业（医生、护士、行政人员）和性别（男、女）的相关关系，则需要先列出列联表：

		职业		
		医生	护士	行政人员
性别	男	a	b	c
	女	d	e	f

其中男医生人数为a，男护士人数为b，男行政人员人数为c，女医生人数为d，女护士人数为e，女行政人员人数为f，该医院总人数（或参与调查的人数）为$a+b+c+d+e+f$。有了列联表，就可以计算一些统计量，如克莱姆相关系数。

列联表又称交叉资料表、交互分类表等，可参见百度百科词条列联表。

5、式(8.40)的解释

当$p_1 = p_2$时，$\kappa = 0$；当$p_1 = 1$时，$\kappa = 1$；一般来说$p_1 \geqslant p_2$，即$\kappa \geqslant 0$，但偶尔也有$p_1 < p_2$的情况，此时$\kappa < 0$。

有关p_1, p_2的意义参见式(8.41)和式(8.42)的解释。

6、式(8.41)的解释

分子$a+d$为分类器h_i与h_j在数据集D上预测结果相同的样本数目，分母为数据集D总样本数目，因此p_1为两个分类器h_i与h_j预测结果相同的概率。

若$a+d=m$，即分类器h_i与h_j对数据集D所有样本预测结果均相同，此时$p_1 = 1$。

7、式(8.42)的解释

将式(8.42)拆分为如下形式，将会很容易理解其含义：

$$p_2 = \frac{a+b}{m} \cdot \frac{a+c}{m} + \frac{c+d}{m} \cdot \frac{b+d}{m}$$

其中$\frac{a+b}{m}$为分类器h_i将样本预测为$+1$的概率，$\frac{a+c}{m}$为分类器h_j将样本预测为$+1$的概率，二者相乘$\frac{a+b}{m} \cdot \frac{a+c}{m}$可理解为分类器$h_i$与$h_j$将样本预测为$+1$的概率；$\frac{c+d}{m}$为分类器$h_i$将样本预测为$-1$的概率，$\frac{b+d}{m}$为分类器$h_j$将样本预测为$-1$的概率，二者相乘$\frac{c+d}{m} \cdot \frac{b+d}{m}$可理解为分类器$h_i$与$h_j$将样本预测为$-1$的概率。

注意$\frac{a+b}{m} \cdot \frac{a+c}{m}$与$\frac{a}{m}$的不同，$\frac{c+d}{m} \cdot \frac{b+d}{m}$与$\frac{d}{m}$的不同：

$$\frac{a+b}{m}\cdot\frac{a+c}{m}=p(h_i=+1)p(h_j=+1),\ \frac{a}{m}=p(h_i=+1,h_j=+1)$$
$$\frac{c+d}{m}\cdot\frac{b+d}{m}=p(h_i=-1)p(h_j=-1),\ \frac{d}{m}=p(h_i=-1,h_j=-1)$$

即$\frac{a+b}{m}\cdot\frac{a+c}{m}$和$\frac{c+d}{m}\cdot\frac{b+d}{m}$是分别考虑分类器$h_i$与$h_j$时的概率（$h_i$与$h_j$独立），而$\frac{a}{m}$和$\frac{d}{m}$是同时考虑$h_i$与$h_j$时的概率（联合概率）。

8、多样性增强的解释

在 8.5.3 节介绍了四种多样性增强的方法，通俗易懂，几乎不需要什么注解，仅强调几个概念：

(1)数据样本扰动中提到了“不稳定基学习器”（例如**决策树**、神经网络等）和“稳定基学习器”（例如线性学习器、支持向量机、朴素贝叶斯、k近邻学习器等），对稳定基学习器进行集成时数据样本扰动技巧效果有限。这也就可以解释为什么随机森林和 GBDT 等以决策树为基分学习器的集成方法很成功吧，Gradient Boosting 和 Bagging 都是以数据样本扰动来增强多样性的；而且，掌握这个经验后在实际工程应用中就可以排除一些候选基分类器，但论文中的确经常见到以支持向量机为基分类器 Bagging 实现，这可能是由于 LIBSVM 简单易用的原因吧。

(2)图 8.11 随机子空间算法，针对每个基分类器h_t在训练时使用了原数据集的部分输入属性（未必是初始属性，详见第 189 页左上注释），因此在最终集成时（图 8.11 最后一行）也要使用相同的部分属性。

(3)输出表示扰动中提到了“翻转法”(Flipping Output)，看起来是一个很无厘头的技巧，为什么要将训练样本的标记改变呢？若认为原训练样本标记是完全可靠的，这不是人为地加入噪声么？但西瓜书作者 2017 年提出的深度森林（参见 IJCAI’17: Deep Forest: Towards an Alternative to Deep Neural Networks）模型中也用到了该技巧，正如本小节名为“多样性增强”，虽然从局部来看引入了标记噪声，但从模型集成的角度来说却是有益的。

8.6、GB(Gradient Boosting)/GBDT/XGBoost

在集成学习中，梯度提升(Gradient Boosting, GB)、梯度提升树(GB Decision Tree, GBDT)很常见，尤其是近几年非常流行的 XGBoost 很是耀眼，此处单独介绍对比这些概念。

1、梯度下降法

本部分内容参考了[孙文瑜, 徐成贤,朱德通.最优化方法[M]. 2 版.北京:高等教育出版社, 2010: 139-140.]。

设目标函数$f(\boldsymbol{x})$在$\boldsymbol{x}_k$附近连续可微，且$\nabla f(\boldsymbol{x}_k)=\frac{\nabla f(\boldsymbol{x})}{\nabla \boldsymbol{x}}\Big|_{\boldsymbol{x}=\boldsymbol{x}_k}\neq 0$。将$f(\boldsymbol{x})$在$\boldsymbol{x}_k$处进行一阶 Taylor 展开

$$f(\boldsymbol{x})\approx f(\boldsymbol{x}_k)+\nabla f(\boldsymbol{x}_k)^{\top}(\boldsymbol{x}-\boldsymbol{x}_k)$$

记$\boldsymbol{x}-\boldsymbol{x}_k=\Delta\boldsymbol{x}$，则上式可写为

$$f(\boldsymbol{x}_k+\Delta\boldsymbol{x})\approx f(\boldsymbol{x}_k)+\nabla f(\boldsymbol{x}_k)^{\top}\Delta\boldsymbol{x}$$

显然，若$\nabla f(\boldsymbol{x}_k)^{\top}\Delta\boldsymbol{x}<0$则有$f(\boldsymbol{x}_k+\Delta\boldsymbol{x})<f(\boldsymbol{x}_k)$，即相比于$f(\boldsymbol{x}_k)$，自变量增量$\Delta\boldsymbol{x}$会使$f(\boldsymbol{x})$函数值下降；若要使$f(\boldsymbol{x})=f(\boldsymbol{x}_k+\Delta\boldsymbol{x})$下降最快，只要选择$\Delta\boldsymbol{x}$使$\nabla f(\boldsymbol{x}_k)^{\top}\Delta\boldsymbol{x}$最小即可，而此时$\nabla f(\boldsymbol{x}_k)^{\top}\Delta\boldsymbol{x}<0$，因此使绝对值$\left|\nabla f(\boldsymbol{x}_k)^{\top}\Delta\boldsymbol{x}\right|$最大即可。将$\Delta\boldsymbol{x}$分成两

部分：$\Delta \boldsymbol{x}=\alpha_k \boldsymbol{d}_k$，其中$\boldsymbol{d}_k$为待求单位向量，$\alpha_k>0$为待解常量；$\boldsymbol{d}_k$表示往哪个方向改变$\boldsymbol{x}$函数值下降最快，而$\alpha_k$表示沿这个方向的步长。因此，求解$\Delta \boldsymbol{x}$的问题变为

$$(\alpha_k, \boldsymbol{d}_k)=\arg\min_{\alpha,\boldsymbol{d}} \nabla f(\boldsymbol{x}_k)^\top \alpha \boldsymbol{d}$$

将以上优化问题分为两步求解，即

$$\boldsymbol{d}_k=\arg\min_{\boldsymbol{d}} \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{d} \quad \text{s.t. } \|\boldsymbol{d}\|_2=1$$

$$\alpha_k=\arg\min_{\alpha} \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{d}_k \alpha$$

以上求解α_k的优化问题明显有问题，因为对于$\nabla f(\boldsymbol{x}_k)^\top \boldsymbol{d}_k<0$来说，显然$\alpha_k=+\infty$时取的最小值，求解$\alpha_k$应该求解如下优化问题：

$$\alpha_k=\arg\min_{\alpha} f(\boldsymbol{x}_k+\alpha \boldsymbol{d}_k)$$

对于凸函数来说，以上两步可以得到最优解；但对于非凸函数来说，联合求解得到$\boldsymbol{d}_k$和α_k，与先求$\boldsymbol{d}_k$然后基于此再求α_k的结果应该有时是不同的。

由 Cauchy-Schwartz 不等式

$$\left|\nabla f(\boldsymbol{x}_k)^\top \boldsymbol{d}_k\right| \leq \|\nabla f(\boldsymbol{x}_k)\|_2 \|\boldsymbol{d}_k\|_2$$

可知，当且仅当$\boldsymbol{d}_k=-\frac{\nabla f(\boldsymbol{x}_k)}{\|\nabla f(\boldsymbol{x}_k)\|_2}$时，$\nabla f(\boldsymbol{x}_k)^\top \boldsymbol{d}_k$最小，$-\nabla f(\boldsymbol{x}_k)^\top \boldsymbol{d}_k$最大。

对于α_k，若$f(\boldsymbol{x}_k+\alpha \boldsymbol{d}_k)$对$\alpha$的导数存在，则可简单求解如下单变量方程即可：

$$\frac{\partial f(\boldsymbol{x}_k+\alpha \boldsymbol{d}_k)}{\partial \alpha}=0$$

例 1：试求$f(x)=x^2$在$x_k=2$处的梯度方向d_k和步长α_k。

解：对$f(x)$在$x_k=2$处进行一阶 Taylor 展开：

$$\begin{aligned} f(x) &= f(x_k)+f'(x_k)(x-x_k) \\ &= x_k^2+2x_k(x-x_k) \\ &= x_k^2+2x_k\alpha d \end{aligned}$$

由于此时自变量为一维，因此只有两个方向可选，要么正方向，要么负方向。此时$f'(x_k)=4$，因此$d_k=-\frac{f'(x_k)}{|f'(x_k)|}=-1$。接下来求$\alpha_k$，将$x_k$和$d_k$代入：

$$f(x_k+\alpha d_k)=f(2-\alpha)=(2-\alpha)^2$$

进而有

$$\frac{\partial f(x_k+\alpha d_k)}{\partial \alpha}=-2(2-\alpha)$$

令导数等于 0，得$\alpha_k=2$。此时

$$\Delta x=\alpha_k d_k=-2$$

则$x_k+\Delta x=0$，函数值$f(x_k+\Delta x)=0$。

例 2：试求$f(\boldsymbol{x})=\|\boldsymbol{x}\|_2^2=\boldsymbol{x}^\top \boldsymbol{x}$在$\boldsymbol{x}_k=[x_k^1, x_k^2]^\top=[3,4]^\top$处的梯度方向$\boldsymbol{d}_k$和步长$\alpha_k$。

解：对$f(\boldsymbol{x})$在$\boldsymbol{x}_k=[x_k^1, x_k^2]^\top=[3,4]^\top$处进行一阶 Taylor 展开：

$$\begin{aligned} f(\boldsymbol{x}) &= f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^\top (\boldsymbol{x} - \boldsymbol{x}_k) \\ &= \|\boldsymbol{x}\|_2^2 + 2\boldsymbol{x}_k^\top (\boldsymbol{x} - \boldsymbol{x}_k) \\ &= \|\boldsymbol{x}\|_2^2 + 2\boldsymbol{x}_k^\top \alpha \boldsymbol{d} \end{aligned}$$

此时$\nabla f(\boldsymbol{x}_k) = [6, 8]^\top$，因此$\boldsymbol{d}_k = -\frac{\nabla f(\boldsymbol{x}_k)}{\|\nabla f(\boldsymbol{x}_k)\|_2} = [-0.6, -0.8]^\top$。接下来求$\alpha_k$，将$\boldsymbol{x}_k$和$\boldsymbol{d}_k$代入：

$$\begin{aligned} f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) &= (3 - 0.6\alpha)^2 + (4 - 0.8\alpha)^2 \\ &= \alpha^2 - 10\alpha + 25 \\ &= (\alpha - 5)^2 \end{aligned}$$

因此可得$\alpha_k = 5$（或对α求导，再令导数等于 0）。此时

$$\Delta \boldsymbol{x} = \alpha_k \boldsymbol{d}_k = [-3, -4]^\top$$

则$\boldsymbol{x}_k + \Delta \boldsymbol{x} = [0, 0]^\top$，函数值$f(\boldsymbol{x}_k + \Delta \boldsymbol{x}) = 0$。

通过以上分析，只想强调两点：

(1)梯度下降法求解下降最快的方向$\boldsymbol{d}_k$时应该求解如下优化问题：

$$\boldsymbol{d}_k = \arg\min_{\boldsymbol{d}} \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{d} \quad \text{s.t. } \|\boldsymbol{d}\|_2 = C$$

其中C为常量，即不必严格限定$\|\boldsymbol{d}_k\|_2 = 1$，只要固定向量长度，与$\alpha_k$搭配即可。

(2)梯度下降法求解步长α_k应该求解如下优化问题：

$$\alpha_k = \arg\min_{\alpha} f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$$

实际应用中，很多时候不会去求最优的α_k，而是靠经验设置一个步长。

2、从梯度下降的角度解释 AdaBoost

AdaBoost 第t轮迭代时最小化式(8.5)的指数损失函数

$$\ell_{\exp}(H_t \mid \mathcal{D}) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[e^{-f(\boldsymbol{x}) H_t(\boldsymbol{x})}] = \sum_{\boldsymbol{x} \in D} \mathcal{D}(\boldsymbol{x}) e^{-f(\boldsymbol{x}) H_t(\boldsymbol{x})}$$

对$\ell_{\exp}(H_t \mid \mathcal{D})$每一项在$H_{t-1}$处泰勒展开

$$\begin{aligned} \ell_{\exp}(H_t \mid \mathcal{D}) &\approx \sum_{\boldsymbol{x} \in D} \mathcal{D}(\boldsymbol{x}) \left(e^{-f(\boldsymbol{x}) H_{t-1}(\boldsymbol{x})} - f(\boldsymbol{x}) e^{-f(\boldsymbol{x}) H_{t-1}(\boldsymbol{x})} (H_t(\boldsymbol{x}) - H_{t-1}(\boldsymbol{x})) \right) \\ &= \sum_{\boldsymbol{x} \in D} \mathcal{D}(\boldsymbol{x}) \left(e^{-f(\boldsymbol{x}) H_{t-1}(\boldsymbol{x})} - e^{-f(\boldsymbol{x}) H_{t-1}(\boldsymbol{x})} f(\boldsymbol{x}) \alpha_t h_t(\boldsymbol{x}) \right) \\ &= \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[e^{-f(\boldsymbol{x}) H_{t-1}(\boldsymbol{x})} - e^{-f(\boldsymbol{x}) H_{t-1}(\boldsymbol{x})} f(\boldsymbol{x}) \alpha_t h_t(\boldsymbol{x})] \end{aligned}$$

其中$H_t = H_{t-1} + \alpha_t h_t$。注意：$\alpha_t, h_t$是第$t$轮待解的变量。

另外补充一下，在上式展开中的变量为$H_t(\boldsymbol{x})$，在H_{t-1}处一阶导数为

$$\left. \frac{\partial e^{-f(\boldsymbol{x}) H_t(\boldsymbol{x})}}{\partial H_t(\boldsymbol{x})} \right|_{H_t(\boldsymbol{x}) = H_{t-1}(\boldsymbol{x})} = -f(\boldsymbol{x}) e^{-f(\boldsymbol{x}) H_{t-1}(\boldsymbol{x})}$$

如果看不习惯上述泰勒展开过程，可令变量$z = H_t(\boldsymbol{x})$和函数$g(z) = e^{-f(\boldsymbol{x}) z}$，对$g(z)$在$z_0 = H_{t-1}(\boldsymbol{x})$处泰勒展开，得

$$\begin{aligned} g(z) &\approx g(z_0)+g'(z_0)(z-z_0) \\ &= g(z_0)-f(\boldsymbol{x})e^{-f(\boldsymbol{x})z_0}(z-z_0) \\ &= e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}-e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}f(\boldsymbol{x})(H_t(\boldsymbol{x})-H_{t-1}(\boldsymbol{x})) \\ &= e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}-e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x}) \end{aligned}$$

注意此处$h_t(\boldsymbol{x})\in\{-1,+1\}$，类似于梯度下降法中的约束$\|\boldsymbol{d}_k\|_2=1$。

类似于梯度下降法求解下降最快的方向$\boldsymbol{d}_k$，此处先求h_t（先不管α_t）：

$$h_t=\arg\min_h \sum_{\boldsymbol{x}\in D}\mathcal{D}(\boldsymbol{x})\left(-e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}f(\boldsymbol{x})h(\boldsymbol{x})\right) \quad \text{s.t. } h(\boldsymbol{x})\in\{-1,+1\}$$

将负号去掉，最小化变为最大化问题

$$\begin{aligned} h_t =& \arg\max_h \sum_{\boldsymbol{x}\in D}\mathcal{D}(\boldsymbol{x})\left(e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}f(\boldsymbol{x})h(\boldsymbol{x})\right) \\ =& \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}f(\boldsymbol{x})h(\boldsymbol{x})] \quad \text{s.t. } h(\boldsymbol{x})\in\{-1,+1\} \end{aligned}$$

这就是式(8.14)的第 3 个等号的结果，因此其余推导参见 8.2 节即可。

由于这里的$h(\boldsymbol{x})$约束较强，因此不能直接取负梯度方向，书中经过推导得到了$h_t(\boldsymbol{x})$的表达式，即式(8.18)。实际上，可以将此结果理解为满足约束条件的最快下降方向。

求得$h_t(\boldsymbol{x})$之后再求α_t（8.2 节“AdaBoost 的个人推导”注解中已经写过一遍，此处仅粘贴至此，具体参见 8.2 节注解，尤其是$\ell_{\exp}(H_{t-1}+\alpha h_t\mid\mathcal{D})$表达式的由来）：

$$\alpha_k=\arg\min_\alpha \ell_{\exp}(H_{t-1}+\alpha h_t\mid\mathcal{D})$$

对指数损失函数$\ell_{\exp}(H_{t-1}+\alpha h_t\mid\mathcal{D})$求导，得

$$\begin{aligned} \frac{\partial\ell_{\exp}(H_{t-1}+\alpha h_t\mid\mathcal{D})}{\partial\alpha} &= \frac{\partial\left(e^{-\alpha}\sum_{i=1}^{|D|}\mathcal{D}_t'(\boldsymbol{x}_i)+(e^{\alpha}-e^{-\alpha})\sum_{i=1}^{|D|}\mathcal{D}_t'(\boldsymbol{x}_i)\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))\right)}{\partial\alpha} \\ &= -e^{-\alpha}\sum_{i=1}^{|D|}\mathcal{D}_t'(\boldsymbol{x}_i)+(e^{\alpha}+e^{-\alpha})\sum_{i=1}^{|D|}\mathcal{D}_t'(\boldsymbol{x}_i)\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i)) \end{aligned}$$

令导数等于零，得

$$\begin{aligned} \frac{e^{-\alpha}}{e^{\alpha}+e^{-\alpha}} =& \frac{\sum_{i=1}^{|D|}\mathcal{D}_t'(\boldsymbol{x}_i)\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))}{\sum_{i=1}^{|D|}\mathcal{D}_t'(\boldsymbol{x}_i)} = \sum_{i=1}^{|D|}\frac{\mathcal{D}_t'(\boldsymbol{x}_i)}{Z_t}\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i)) \\ =& \sum_{i=1}^{|D|}\mathcal{D}_t(\boldsymbol{x}_i)\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i)) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[\mathbb{I}(f(\boldsymbol{x}_i)\neq h(\boldsymbol{x}_i))\right] \\ =& \epsilon_t \end{aligned}$$

对上述等式化简，得

$$\begin{aligned} &\frac{e^{-\alpha}}{e^{\alpha}+e^{-\alpha}}=\frac{1}{e^{2\alpha}+1}\Rightarrow e^{2\alpha}+1=\frac{1}{\epsilon_t}\Rightarrow e^{2\alpha}=\frac{1-\epsilon_t}{\epsilon_t}\Rightarrow 2\alpha=\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \\ &\Rightarrow \alpha_t=\frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \end{aligned}$$

即式(8.11)。

通过以上推导可以发现：AdaBoost 每一轮的迭代就是基于梯度下降法求解损失函数为指数损失函数的二分类问题（约束条件$h_t(\boldsymbol{x})\in\{-1,+1\}$）。

3、梯度提升(Gradient Boosting)

将 AdaBoost 的问题一般化，即不限定损失函数为指数损失函数，也不局限于二分类问题，则可以将式(8.5)写为更一般化的形式

$$\begin{aligned}\ell(H_t \mid \mathcal{D}) =&\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\mathrm{err}(H_t(\boldsymbol{x}), f(\boldsymbol{x}))] \\ =&\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\mathrm{err}(H_{t-1}(\boldsymbol{x}) + \alpha_t h_t(\boldsymbol{x}), f(\boldsymbol{x}))]\end{aligned}$$

易知，当$f(\boldsymbol{x}) \in \{-1, +1\}$且$\mathrm{err}(H_t(\boldsymbol{x}), f(\boldsymbol{x})) = e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}$时，就是式(8.5)。当是回归问题时，$f(\boldsymbol{x}) \in \mathbb{R}$，损失函数可使用平方损失$\mathrm{err}(H_t(\boldsymbol{x}), f(\boldsymbol{x})) = (H_t(\boldsymbol{x}) - f(\boldsymbol{x}))^2$。

针对该一般化的损失函数和一般的学习问题，要通过T轮迭代得到学习器

$$H(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$$

类似于 AdaBoost，第t轮得到$\alpha_t, h_t(\boldsymbol{x})$，可先对损失函数在$H_{t-1}(\boldsymbol{x})$处进行泰勒展开：

$$\begin{aligned}\ell(H_t \mid \mathcal{D}) \approx&\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\mathrm{err}(H_{t-1}(\boldsymbol{x}), f(\boldsymbol{x})) + \left.\frac{\partial \mathrm{err}(H_t(\boldsymbol{x}), f(\boldsymbol{x}))}{\partial H_t(\boldsymbol{x})}\right|_{H_t(\boldsymbol{x})=H_{t-1}(\boldsymbol{x})} (H_t(\boldsymbol{x}) - H_{t-1}(\boldsymbol{x}))\right] \\ =&\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\mathrm{err}(H_{t-1}(\boldsymbol{x}), f(\boldsymbol{x})) + \left.\frac{\partial \mathrm{err}(H_t(\boldsymbol{x}), f(\boldsymbol{x}))}{\partial H_t(\boldsymbol{x})}\right|_{H_t(\boldsymbol{x})=H_{t-1}(\boldsymbol{x})} \alpha_t h_t(\boldsymbol{x})\right] \\ =&\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\mathrm{err}(H_{t-1}(\boldsymbol{x}), f(\boldsymbol{x}))\right] + \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\left.\frac{\partial \mathrm{err}(H_t(\boldsymbol{x}), f(\boldsymbol{x}))}{\partial H_t(\boldsymbol{x})}\right|_{H_t(\boldsymbol{x})=H_{t-1}(\boldsymbol{x})} \alpha_t h_t(\boldsymbol{x})\right]\end{aligned}$$

注意，在上式展开中的变量为$H_t(\boldsymbol{x})$，且有$H_t(\boldsymbol{x}) = H_{t-1}(\boldsymbol{x}) + \alpha_t h_t(\boldsymbol{x})$（类似于梯度下降法中$\boldsymbol{x} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$）。上式中括号内第 1 项为常量$\ell(H_{t-1} \mid \mathcal{D})$，最小化$\ell(H_t \mid \mathcal{D})$只须最小化第 2 项即可。先不考虑权重$\alpha_t$，求解如下优化问题可得$h_t(\boldsymbol{x})$：

$$h_t(\boldsymbol{x}) = \arg\min_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\left.\frac{\partial \mathrm{err}(H_t(\boldsymbol{x}), f(\boldsymbol{x}))}{\partial H_t(\boldsymbol{x})}\right|_{H_t(\boldsymbol{x})=H_{t-1}(\boldsymbol{x})} h(\boldsymbol{x})\right] \quad \text{s.t. constraints for } h(\boldsymbol{x})$$

解得$h_t(\boldsymbol{x})$之后，再求解如下优化问题可得权重α_t：

$$\alpha_t = \arg\min_{\alpha} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\mathrm{err}(H_{t-1}(\boldsymbol{x}) + \alpha h_t(\boldsymbol{x}), f(\boldsymbol{x}))]$$

以上就是梯度提升(Gradient Boosting)的理论框架，即每轮通过梯度(Gradient)下降的方式将T个弱学习器提升(Boosting)为强学习器。可以看出 AdaBoost 是其特殊形式。

Gradient Boosting 算法的官方版本参见[Friedman J H. Greedy function approximation: a gradient boosting machine[J]. Annals of statistics, 2001: 1189-1232.]的第 5-6 页(第 1193-1194 页)：

ALGORITHM 1 (Gradient_Boost).

1. $F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} L(y_i, \rho)$
2. For $m = 1$ to M do:
3. $\tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})},\ i = 1, N$
4. $\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$
5. $\rho_m = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$
6. $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$

7. endFor
end Algorithm

感觉该伪代码针对的还是在任意损失函数$L(y_i, F(\boldsymbol{x}_i))$下的回归问题。Algorithm 1 中第 3 步和第 4 步意思是用$\beta h(\boldsymbol{x}_i, \boldsymbol{a})$拟合$F(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x})$处负梯度，但第 4 步表示只求参数$\boldsymbol{a}_m$，第 5 步单独求解参数$\rho_m$，这里的疑问是为什么第 4 步要用最小二乘法（即 3.2 节的线性回归）去拟合负梯度（又称伪残差）？

简单理解如下：第 4 步要解的$h(\boldsymbol{x}_i, \boldsymbol{a})$相当于梯度下降法中的待解的下降方向$\boldsymbol{d}$，在梯度下降法中也已提到不必严格限制$\|\boldsymbol{d}\|_2 = 1$，长度可以由步长$\alpha$调节（例如前面梯度下降方解释中的例 1，若直接取$d_k = -f'(x_k) = -4$，则可得$\alpha_k = 0.5$，仍有$\Delta x = \alpha_k d_k = -2$），因此第 4 步直接用$h(\boldsymbol{x}_i, \boldsymbol{a})$拟合负梯度，与梯度下降中约束$\|\boldsymbol{d}\|_2 = 1$的区别在于未对负梯度除以其模值进行归一化而已。

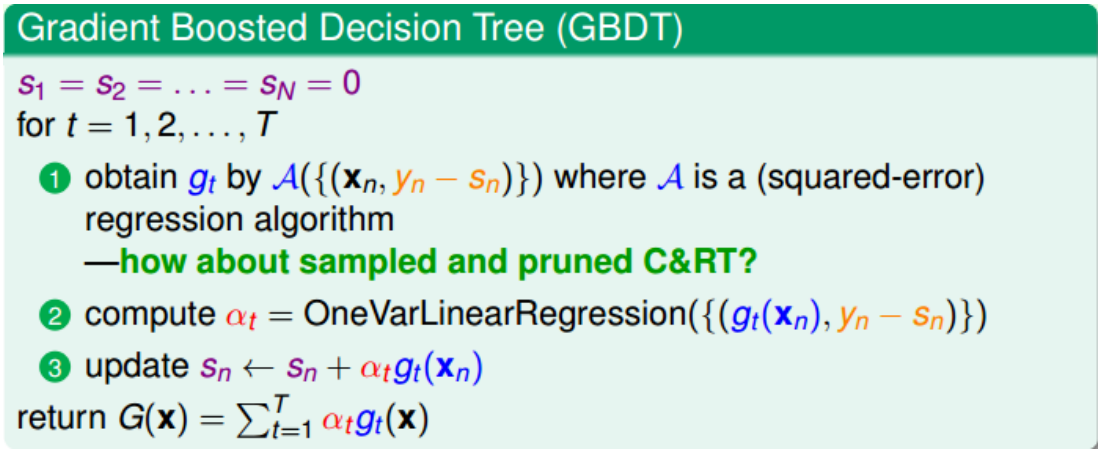
那为什么不是直接令$h(\boldsymbol{x}_i, \boldsymbol{a})$等于负梯度呢？因为这里实际是求假设函数$h$，将数据集中所有的$\boldsymbol{x}_i$经假设函数$h$映射到对应的伪残差（负梯度）$\tilde{y}_i$，所以只能做线性回归了。

李航《统计学习方法》第 8.4.3 节中的算法 8.4 并未显式体现参数ρ_m，这应该是第 2 步的(c)步完成的，因为(b)步只是拟合一棵回归树（相当于 Algorithm 1 第 4 步解得$h(\boldsymbol{x}_i, \boldsymbol{a})$），而(c)步才确定每个叶结点的取值（相当于 Algorithm 1 第 5 步解得ρ_m，只是每个叶结点均对应一个ρ_m）；而且回归问题中基函数为实值函数，可以将参数ρ_m吸收到基函数中。

4、梯度提升树(GBDT)

本部分无实质 GBDT 内容，仅为梳理 GBDT 的概念，具体可参考给出的资源链接。

对于 GBDT，一般资料是按 Gradient Boosting+CART 处理回归问题讲解的，如林轩田《机器学习技法》课程第 11 讲（损失函数为平方损失，PPT 中的 C&RT 是 CART 的基础版本）：



但是，分类问题也可以用回归来处理，例如 3.3 节的对数几率回归，只需将平方损失换为对率损失（参见式(3.27)和式(6.33)，二者关系可参见第 3 章注解中有关式(3.27)的推导）即可。博客梯度提升树(GBDT)原理小结(https://www.cnblogs.com/pinard/p/6140514.html)从更一般的角度推导了 GBDT，包括 GBDT 回归版本和分类版本。

特别地，林轩田老师（个人主页：https://www.csie.ntu.edu.tw/~htlin/）的《机器学习基石》和《机器学习技法》两门课程的主页参见 https://www.csie.ntu.edu.tw/~htlin/mooc/，课程视频 https://www.bilibili.com/video/av12463015/和 https://www.bilibili.com/video/av12469267/。

5、XGBoost

本部分无实质 XGBoost 内容，仅为梳理 XGBoost 的概念，具体可参考给出的资源链接。

首先，XGBoost 是 e**X**treme **G**radient **Boost**ing 的简称。

其次，XGBoost 与 GBDT 的关系，可大致类比为 LIBSVM 与 SVM（或 SMO 算法）的关系。LIBSVM 是 SVM 算法的一种高效实现软件包，XGBoost 是 GBDT 的一种高效实现；在实现层面，LIBSVM 对 SMO 算法进行了许多改进，XGBoost 也对 GBDT 进行了许多改进；另外，LIBSVM 扩展了许多 SVM 变体，XGBoost 也不再仅仅是标准的 GBDT，也扩展了一些其它功能。

最后，XGBoost 是由陈天奇（个人主页：https://homes.cs.washington.edu/~tqchen/）开发的；XGBoost 论文发表在 KDD'16 上，可在 arXiv 下载 https://arxiv.org/abs/1603.02754；XGBoost 工具包主页 https://xgboost.ai/，文档 https://xgboost.readthedocs.io/en/latest/，源码 https://github.com/dmlc/xgboost；统计之都(https://cosx.org/)曾对陈天奇做过一个采访，参见《COS 访谈第 18 期：陈天奇》(https://cosx.org/2015/06/interview-of-tianqi)，里面介绍的一些有关作者的信息是论文中看不到的。

官方文档开篇如是介绍 XGBoost：

> **XGBoost** is an optimized distributed gradient boosting library designed to be highly **efficient**, **flexible** and **portable**. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

GB、GBDT、XGBoost 的递进关系简单介绍可参见：一步一步理解 GB、GBDT、xgboost (https://www.cnblogs.com/wxquare/p/5541414.html)；另外，有关 XGBoost 介绍，很多人推荐陈天奇的 PPT：https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf。

特别崇拜一下 XGBoost 作者陈天奇，引用新浪微博 Maybe_Shewill 的评价："撸得了底层框架又搞得了上层高端算法的大神"。

8.7 本章小节

本章内容整体上看并不复杂，可以看成是作者的英文专著《Ensemble Methods: Foundations and Algorithms》（https://book.douban.com/subject/10494228/，据悉该书中文版即将由电子工业出版社出版）的精简版，但 8.2 节有关 AdaBoost 的推导感觉很难吃透，尤其是本节还涉及到如 Gradient Boosting、GBDT、XGBoost 等概念，因此必须读一些原始文献和其它参考资料方能对此有所理解，下面对这些概念进行一个简单的梳理。

提升(Boosting)是一族可将弱学习提升为强学习器的算法，弱学习器间存在强依赖关系、必须串行生成。这族算法的工作机制类似：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，然后基于调整后打着本分布来训练下一个基学习器；如此重复进行，直至基学习器数目达到事先指定的值 T，最终将这 T 个基学习器进行加权结合。（摘自西瓜书 P173）

提升树(Boosting Tree)是以决策树为基学习器的提升方法(i.e., Boosting + Decision Tree)。对分类问题决策树是二叉分类树，对回归问题决策树是二叉回归树。(李航《统计学习方法》)

梯度提升(Gradient Boosting)是基于加法模型和前向分步算法一种理论框架，并不限定损失函数，也不限定基分类器的类型（类型包含两层含义：第 1 层如分类和回归等；第 2 层如决策树、SVM、神经网络等）。

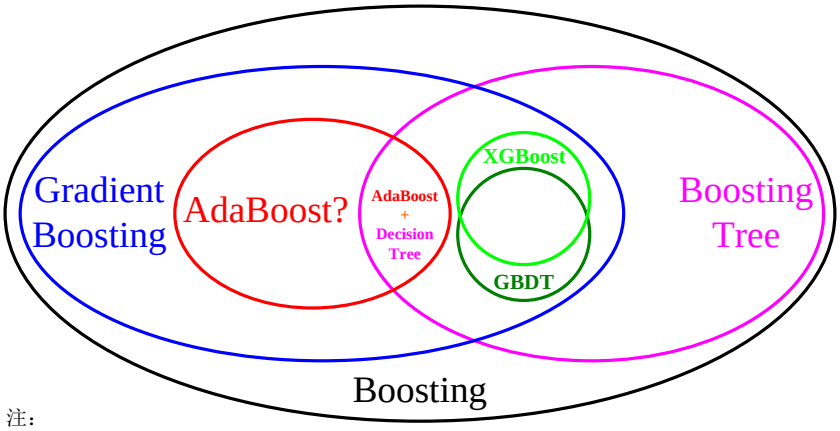
梯度提升树(GBDT) = Gradient Boosting + Decision Tree (CART)，这里 CART 取为回归树。李航《统计学习方法》第 8.4.3 节中的算法 8.4 实际为 GBDT 算法，与刘建平 Pinard 博客《梯度提升树(GBDT)原理小结》介绍基本一致，林轩田《机器学习技法》课程第 11 讲则特指损失函数为平方损失。

特别地，8.2 节介绍的 **AdaBoost(Adaptive Boosting)**是当 Gradient Boosting 中损失函数为指数损失函数、基分类器为二分类（只能取+1 和-1 两个值）时的特殊形式(i.e., Gradient Boosting + Exponential Loss + Binary base classifier)，这里基分类器可以为任何二分类器，如决策树、SVM、神经网络等。当然，西瓜书作者在第 190 页的阅读材料中提到，这仅是源“统计视角”的一种推导而已，但这派理论产生的推论并不能解释某些现象，因此这可能仅是一个与 AdaBoost 相似的学习过程而非 AdaBoost 本身。

当 AdaBoost 的基分类器为二分类决策树时，也可以称为**提升树**，“对二类分类问题，提升树算法只需将 AdaBoost 算法 8.1 中的基本分类器限制为二类分类树即可，可以说这时的提升树是 AdaBoost 算法的特殊情况”。但这里与 GBDT 不一样，因为 AdaBoost 限定了基分类器为二分类（实际上也可以做回归任务），并不像 GBDT 里那样，每轮迭代与残差（负梯度）拟合；而且已经提到，GBDT 一般特指 Gradient Boosting 与 CART 回归树的结合。

XGBoost 是 GBDT 的一个软件包，类似于 LIBSVM 是 SVM 的一个软件包。XGBoost 在实现 GBDT 时进行了一些细节上的改进和扩展。

各种概念之间的包含关系如下图所示：



注：
(1)以上范围重叠仅表示两个概念有交集，但重叠的面积大小与概念的重叠程度不成比例;
(2)西瓜书第190页的阅读材料提到8.2节的推导可能仅是一个与AdaBoost相似的学习过程而非AdaBoost本身，因此图中添加了问号；
(3)XGBoost是GBDT的一种实现，但也有改进和扩展，因此图中二者范围并未完全重叠；

以上仅为个人的初步理解，可能有误，勿被误导^_^