第七讲 集成学习

牟克典

2025年3月

概要

集成学习：通过将相对比较容易构建但泛化性能一般的多个学习器进行结合，积弱成强，使得泛化能力得到显著提升.

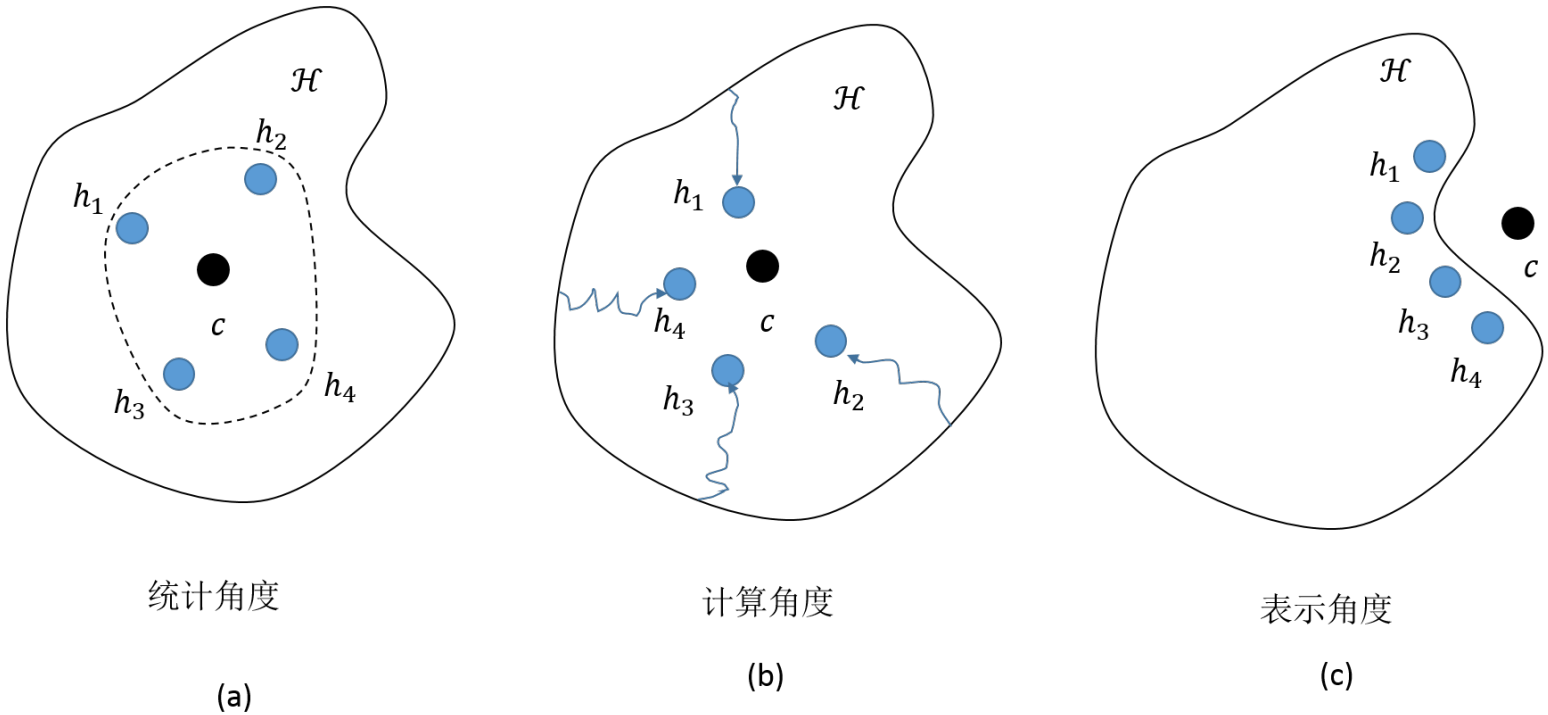

Figure: 集成表现更好的三个理由[Thomas G. Dietterich2000]

- 个体学习器的学习与多个学习器的结合机制是集成学习的两个重要环节.
- 根据集成学习中个体分类器是否由同一学习算法学得将集成区分为同质的和异质的:
  - 所谓同质的集成学习是指每个个体学习器都是由同一学习算法学得，称相应的个体学习器为基学习器，相应的学习算法为基学习算法;
  - 反之称集成是异质的，并且称相应的个体学习器为组件学习器.
- 根据个体学习器的学习之间是否存在依赖关系，将集成学习方法分为序列化方法和并行化方法:
  - 所谓序列化方法是指待结合的个体学习器的学习之间存在依赖关系、必须串行学得，个体学习器构成一个序列，如提升方法;
  - 并行化方法中待结合的个体学习器的学习之间无相互依赖关系，因此各个体学习器的生成无特定次序要求，可同时生成，如Bagging和随机森林.

概要

- 提升方法是将弱学习算法提升为强学习算法的方法.
- 从理论上来说，在PAC框架下概念类是强可学习的当且仅当这个概念类是弱可学习的.
- 但在实际任务中，发现弱学习算法相对于发现强学习算法通常要更容易一些.
- 将一个弱学习算法提升为强学习算法有意义.
- 典型的提升方法:
  - AdaBoost算法:
    - 针对二分类问题.
    - 采用加权多数表决的方式来对若干个依次学到的同质的弱分类器进行集成.
  - 提升树模型:
    - 以回归树作为基本分类器，以加法模型的形式实现对基本分类器的集成.

AdaBoost算法

- 给定训练数据集$D = \{(x_i, y_i)\}_{i=1}^N$，其中$x_i \in \mathcal{X}$，$y_i \in \mathcal{Y} = \{-1, +1\}$, $1 \leq i \leq N$.
- 用$\mathcal{L}$表示AdaBoost算法中所采用的弱学习算法.
- T表示需要学习的基本分类器的个数.
- 以$W_t = \{w_{i,t}\}_{i=1}^N$表示AdaBoost算法中第$t$轮学习时训练数据集$D$上的权值分布，其中
  - $w_{i,t}$为x_i的权值.
  - 通常x_i的初始权值$w_{i,1} = \frac{1}{N}$.
- AdaBoost算法在第t轮的学习中应用算法$\mathcal{L}$基于训练数据集D和D上的权值分布W_t学得具有最小训练误差的基分类器$f_t(x)$，即

$$f_t = \operatorname*{argmin}_f \sum_{i=1}^N w_{i,t} I(f(x_i) \neq y_i).$$

- 计算$f_t(x)$在训练样本集上的分类错误率

$$e_t = \sum_{i=1}^{N} w_{i,t} I(f_t(x_i) \neq y_i).$$

- 基于分类错误率e_t，计算$f_t(x)$在加权多数投票集成机制中的权值

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

- 同时，采用如下方法将D上的权值分布W_t更新为W_{t+1}，用于下一轮的学习：

$$w_{i,t+1} = \frac{w_{i,t}}{Z_t} \exp(-\alpha_t y_i f_t(x_i)),$$

其中$Z_t = \sum_{i=1}^{N} w_{i,t} \exp(-\alpha_t y_i f_t(x_i))$ 为使得所有$w_{i,t+1}$之和为1的规范化因子.

- 经过T轮学习以后，将$\{f_1(x), f_2(x), \cdots, f_T(x)\}$按照如下方式集成得到$f$:

$$f(x) = \text{sign}(G(x)),$$

其中

$$G(x) = \sum_{t=1}^{T} \alpha_t f_t(x).$$

- 对这种集成方式来说，$G(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$的符号决定了$x$的类别标记，而$|G(x)| = |\sum_{t=1}^{T} \alpha_t f_t(x)|$给出了$f$对$x$的类别预测的确信度.

关于AdaBoost算法

- 在$e_t < \frac{1}{2}$时，$\alpha_t > 0$，而且e_t越小，α_t越大，这说明在组合基分类器的时候对性能越好的基分类器赋以更大的权重.
- α_t还通过如下方式在样本分布权值的调整中起作用：

$$w_{i,t+1} = \begin{cases} \frac{w_{i,t}}{Z_t}\exp(-\alpha_t), & y_i = f_t(x_i), \\ \frac{w_{i,t}}{Z_t}\exp(\alpha_t), & y_i \neq f_t(x_i) \end{cases}$$

即

  - 对误分类样本通过乘以$\exp(\alpha_t)$提高其权值比重.
  - 对被正确分类的样本通过乘以$\exp(-\alpha_t)$降低其权值比重.

关于AdaBoost算法

- 从α_t的定义可以得到

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

由此进一步得到

$$\exp(2\alpha_t)e_t = 1 - e_t.$$

上式两边各乘以$\exp(-\alpha_t)$就可得到

$$\exp(\alpha_t)e_t = \exp(-\alpha_t)(1 - e_t).$$

- 这说明在每次更新权值分布过程中,分配给误分类样本的权值之和与分配给被正确分类样本的权值之和相等.

关于AdaBoost算法

- 规范化因子Z_t可以完全由e_t表示：

$$\begin{aligned} Z_t &= \sum_{i=1}^{N} w_{i,t} \exp(-\alpha_t y_i f_t(x_i)) \\ &= \sum_{y_i \neq f_t(x_i)} w_{i,t} \exp(\alpha_t) + \sum_{y_i = f_t(x_i)} w_{i,t} \exp(\alpha_t) \\ &= \exp(\alpha_t) e_t + \exp(-\alpha_t)(1 - e_t). \end{aligned}$$

考虑式$\exp(\alpha_t) e_t = \exp(-\alpha_t)(1 - e_t)$可得

$$\begin{aligned} Z_t &= 2 \exp(\alpha_t) e_t \\ &= 2 \left[(\frac{1 - e_t}{e_t})^{\frac{1}{2}} e_t \right] = 2\sqrt{e_t(1 - e_t)}. \end{aligned}$$

关于AdaBoost算法

- 权值调整累积过程:

$$w_{i,t+1} = w_{i,t}\frac{\exp(-y_i\alpha_t f_t(x_i))}{Z_t}$$
$$= w_{i,1}\frac{\exp(-y_i(\sum_{s=1}^{t}\alpha_s f_s(x_i)))}{\prod_{s=1}^{t} Z_s}$$
$$= \frac{1}{N}\cdot\frac{\exp(-y_i(\sum_{s=1}^{t}\alpha_s f_s(x_i)))}{\prod_{s=1}^{t} Z_s}.$$

特别地，当$t = T$时，

$$w_{i,T+1} = \frac{1}{N}\cdot\frac{\exp(-y_i(\sum_{s=1}^{T}\alpha_s f_s(x_i)))}{\prod_{s=1}^{T} Z_s}$$
$$= \frac{1}{N}\cdot\frac{\exp(-y_i G(x_i))}{\prod_{s=1}^{T} Z_s}.$$

关于AdaBoost算法

- 损失、权值与规范化因子
  由

$$w_{i,T+1} = \frac{1}{N} \cdot \frac{\exp(-y_i G(x_i))}{\prod_{s=1}^{T} Z_s}.$$

  可以得到

$$\exp(-y_i G(x_i)) = N \cdot w_{i,T+1} (\prod_{t=1}^{T} Z_t).$$

AdaBoost算法的基本框架

AdaBoost算法

输入： 训练样本集$D = \{(x_i, y_i)\}_{i=1}^N$；弱分类学习算法$\mathcal{L}$；基分类器个数$T$

输出： 集成分类器$f(x)$

1 初始化$D = \{(x_i, y_i)\}_{i=1}^N$上的权值分布$W_1 = \{w_{i,1} = \frac{1}{N}\}_{i=1}^N$.

2 对$t = 1, 2, \ldots, T$

(a) 从(D, W_t)依学习算法$\mathcal{L}$学得基分类器$f_t(x)$.

(b) 计算$f_t(x)$在(D, W_t)上的分类误差率e_t.

(c) 计算$f_t(x)$的权重系数α_t.

(d) 计算权值分布更新规范化因子

$$Z_t = 2\sqrt{e_t(1 - e_t)}.$$

(e) 将D上的权值分布更新为

$$W_{t+1} = \{w_{i,t+1} = \frac{w_{i,t}}{Z_t} \exp(-\alpha_t y_i f_t(x_i))\}_{i=1}^N.$$

AdaBoost算法的基本框架

3 构建基分类器

$$\{f_t(x)\}_{t=1}^{T}$$

的线性组合

$$G(x) = \sum_{t=1}^{T} \alpha_t f_t(x).$$

4 返回集成分类器

$$f(x) = \text{sign}(G(x)).$$

- $e_t < \frac{1}{2}$是α_t和相应的权值调整因子$\exp(-\alpha_t)$和$\exp(\alpha_t)$ 有意义的前提条件.
- 如果出现分类误差率$e_t \geq \frac{1}{2}$的情况, 如何处理?

训练误差上界

集成分类器f的训练误差为：

$$\hat{R}(f) = \frac{1}{N}\sum_{i=1}^{N} I(y_i \neq f(x_i)).$$

从集成分类器的构建可知，$y_i \neq f(x_i)$ 等价于 $y_i G(x_i) \leq 0$，因此f的训练误差也可以表示为

$$\hat{R}(f) = \frac{1}{N}\sum_{i=1}^{N} I(y_i G(x_i) \leq 0).$$

由于$I(y_i G(x_i) \leq 0) \leq \exp(-y_i G(x_i))$,则

$$\hat{R}(f) \leq \frac{1}{N}\sum_{i=1}^{N} \exp(-y_i G(x_i)) = \frac{1}{N}\sum_{i=1}^{N} \exp(-y_i(\sum_{t=1}^{T} \alpha_t f_t(x_i))).$$

回顾权值更新公式

$$w_{i,T+1} = \frac{1}{N} \cdot \frac{\exp(-y_i G(x_i))}{\prod_{s=1}^{T} Z_s},$$

我们得到

$$\begin{aligned} & \frac{1}{N} \sum_{i=1}^{N} \exp(-y_i (\sum_{t=1}^{T} \alpha_t f_t(x_i))) \\ = & \frac{1}{N} \sum_{i=1}^{N} [N \prod_{t=1}^{T} Z_t w_{i,T+1}] = \prod_{t=1}^{T} Z_t. \end{aligned}$$

将式$Z_t = 2\sqrt{e_t(1-e_t)}$代入上式，我们可以得到

$$\begin{aligned}\prod_{t=1}^{T} Z_t &= \prod_{t=1}^{T} 2\sqrt{e_t(1-e_t)} \\ &= \prod_{t=1}^{T} \sqrt{1-4(\frac{1}{2}-e_t)^2}.\end{aligned}$$

令$\gamma_t = \frac{1}{2} - e_t$，则

$$\begin{aligned}\prod_{t=1}^{T} \sqrt{1-4(\frac{1}{2}-e_t)^2} &= \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2} \\ &\leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).\end{aligned}$$

因此我们得到

$$\hat{R}(f) \leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).$$

特别地，如果对每个γ_t都有$\gamma \leq \gamma_t$，则

$$\hat{R}(f) \leq \exp\left(-2T\gamma^2\right).$$

这说明

- AdaBoost算法的训练误差是以负指数量级下降的.
- 而且在应用AdaBoost算法的时候我们并不需要提前知道γ_t和γ，这也是AdaBoost算法名字中Ada代表的Adaptive的含义所在.

加法模型

AdaBoost算法构建的基分类器的线性组合

$$G(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$$

正好是一个加法模型，AdaBoost算法也符合前向分步算法的特点:

- 把$G(x)$看作一个参数为

$$\{(\alpha_1, f_1), (\alpha_2, f_2), \cdots, (\alpha_T, f_T)\}$$

的加法模型，

- AdaBoost算法相当于采用前向分步的方式依次确定(α_t, f_t).

如果我们在训练误差中采用指数损失函数$\exp(-yf(x))$, 设初始模型为$f_0 = 0$, 采用前向分步的方式在$t-1$步（轮）得到的模型为

$$G_{t-1}(x) = \sum_{s=0}^{t-1} \alpha_s f_s(x).$$

令

$$G_t(\alpha, f) = \frac{1}{N} \sum_{i=1}^{N} \exp(-y_i(G_{t-1}(x_i) + \alpha f(x_i))),$$

则可以证明在t步（轮）得到的(α_t, f_t)满足

$$(\alpha_t, f_t) = \underset{(\alpha,f)}{\operatorname{argmin}}\, G_t(\alpha, f).$$

注意到

$$G_t(\alpha, f) = \frac{1}{N}\sum_{i=1}^{N}\exp(-y_i G_{t-1}(x_i))\exp(-y_i\alpha f(x_i)))$$

$$= \frac{1}{N}\sum_{i=1}^{N}\exp(-y_i(\sum_{s=0}^{t-1}\alpha_s f_s(x_i))\exp(-y_i\alpha f(x_i)))$$

$$= \prod_{s=1}^{t-1} Z_s(\sum_{i=1}^{N} w_{i,t}\exp(-y_i\alpha f(x_i)))$$

$$= \prod_{s=1}^{t-1} Z_s(\sum_{y_i=f(x_i)} w_{i,t}\exp(-\alpha) + \sum_{y_i\neq f(x_i)} w_{i,t}\exp(\alpha)).$$

令$e_t = \sum_{y_i \neq f(x_i)} w_{i,t}$, 则

$$G_t(\alpha, f) = \prod_{s=1}^{t-1} Z_s \left((1 - e_t)\exp(-\alpha) + e_t \exp(\alpha)\right)$$

$$= \prod_{s=1}^{t-1} Z_s \left(e_t(\exp(\alpha) - \exp(-\alpha)) + \exp(-\alpha)\right).$$

显然$\prod_{s=1}^{t-1} Z_s$既与α无关,也与f无关,要使得$G_t(\alpha, f)$最小化,对任意的$\alpha > 0$来说,需要e_t最小化,即

$$f_t(x) = \operatorname{argmin}_f \sum_{y_i \neq f(x_i)} w_{i,t}.$$

令$G_t(\alpha, f)$对α的偏导数$\frac{\partial G_t(\alpha,f)}{\partial \alpha}$为0，则得到

$$-(1-e_t)\exp(-\alpha)+e_t\exp(\alpha)=0,$$

进一步可得

$$\alpha=\frac{1}{2}\ln\frac{(1-e_t)}{e_t}.$$

这与AdaBoost算法完全一致.

提升树模型

- 以分类或者回归树作为基学习器
- 以基学习器的线性组合来作为集成机制构造的如下加法模型：

$$f(x) = \sum_{m=1}^{M} T(x; \Theta_m),$$

其中

- 每个$T(x; \Theta_m)$都是以Θ_m为参数的决策树(分类或者回归树)模型，
- M为基学习器的个数.

- 给定训练数据集$D = \{(x_i, y_i)\}_{i=1}^N$，若损失函数为$L(y, f(x))$，则第$m$个基学习器$T(x; \Theta_m)$由如下经验风险最小化原则学得：

$$\Theta_m = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i; \Theta)).$$

- 特别地，我们考虑基学习器为回归树、损失函数为平方损失$L_2(y, f(x)) = (y - f(x))^2$的情形，则

$$\Theta_m = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^{N} [y_i - (f_{m-1}(x_i) + T(x_i; \Theta))]^2.$$

- 定义当前模型拟合数据的残差为

$$r_{mi} = y_i - f_{m-1}(x_i), \quad i = 1, 2, \ldots, N.$$

则

$$\Theta_m = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^{N} (r_{mi} - T(x_i; \Theta))^2.$$

- 提升树算法每轮基学习器的学习就是在拟合当前模型的残差.

概要

- 对训练样本进行重采样，利用不同的样本数据来学习不同的基学习器，通过降低方差来提高集成学习器的泛化性能.
- Bagging (**Bootstrap AGGregatING**)算法就是这类算法的代表之一。

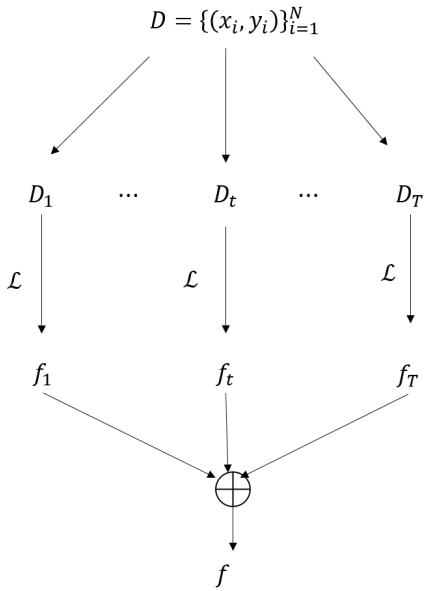

Figure: Bagging示意

- Bagging算法每次从给定的容量为N的训练数据集D中利用自助采样法(Bootstrap Sampling)随机抽取N个样本得到重采样样本集D_t,
- 然后依基学习算法$\mathcal{L}$学得一个基学习器f_t,
- 经过T轮学习之后得到T个基学习器$\{f_1, f_2, \cdots, f_T\}$.
- 对于分类问题，Bagging 算法使用多数占优的简单投票法作为这些基分类器的结合机制.
- 对于回归问题，Bagging 算法以这些基学习器的回归值的均值作为集成学习器的回归值.

针对分类任务的Bagging算法

Bagging算法

输入： 训练样本集$D = \{(x_i, y_i)\}_{i=1}^{N}$，$x_i \in \mathcal{X}$，$y_i \in \mathcal{Y}$，$i = 1, 2, \ldots, N$；弱分类算法$\mathcal{L}$；基分类器个数$T$

输出： 集成分类器$f(x)$

1. 对$t = 1, 2, \ldots, T$
   (a) 从D利用自助采样法随机抽取N个样本得到D_t。
   (b) 从D_t依学习算法$\mathcal{L}$学得基分类器$f_t(x)$。
2. 返回集成分类器

$$f(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{t=1}^{T} I(f_t(x) = y).$$

- 利用自助采样法可以认为D中约有63.2%的样本出现在D_t中,这在一定程度上有助于避免D_t的代表性不足问题.
- 同时对每一轮采样来说，约有36.8%的样本不出现在D_t中，这也有助于各轮D_t相互之间保持比较大的差异.
- 对样本x_i 来说，我们采用x_i未参与训练的基学习器在x_i 上的预测的简单投票结果作为x_i 的包外预测

$$f_{oob}(x_i) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{t=1}^{T} I(f_t(x_i) = y) I(x_i \notin D_t),$$

- 相应的Bagging算法泛化误差的包外估计(out-of-bag estimate)为

$$\epsilon^{oob} = \frac{1}{N} \sum_{i=1}^{N} I(f_{oob}(x_i) \neq y_i).$$

概要

随机森林模型

- 在Bagging方法中以决策树算法作为基学习器的学习算法
- 并在决策树的学习算法中引入随机属性选择.

所谓随机属性选择是指

- 决策树算法在选择划分特征时先从当前结点对应的所有特征（不妨设共d有个特征）中随机选择k 个特征作为候选划分特征
- 再从这k 个特征中选择最优划分特征.

- 如果$k = d$，则此情形下的随机森林模型就是基学习算法采用决策树算法的Bagging.
- 如果$k < d$，相当于在特征选择中及其对应的数据集的划分方面引入随机扰动.
- 特别在$k = 1$的极端情况下，相当于每次随机选择一个特征用于划分.
- 参数k的值在实际应用中一般推荐$k = \log_2 d$.

- 与Bagging方法相比，随机属性选择使得随机森林模型多了一层关于划分特征的随机扰动.
- 这有助于增加基学习器的多样性差异，进而提升最终集成学习器的泛化能力.
- 另一方面，随机森林的基学习算法在每次划分结点的时候只随机考察一个规模为k的特征子集.
- 这使得每个基分类器的训练效率高，但性能相对比无属性扰动的情形有所降低.
- 但随着基学习器数目的增加，随机森林通常会收敛到比Bagging更低的泛化误差.

小结

- 集成学习
- 提升方法
- AdaBoost算法
- 提升树
- Bagging方法
- 随机森林