机器学习：集成学习(Ensemble Learning)

~~Copyright: Jingmin Wei, Automation - Pattern Recognition and Intelligent System, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology~~

Copyright: Jingmin Wei, Computer Science - Artificial Intelligence, Department of Computer Science, Viterbi School of Engineering, University of Southern California

---

---

1. 概述

案例：对于不同的癌症症状，不同教授的预测准确率不尽相同(给出的预测概率)

定义：通过组合多种模型和方法，集成学习能够有效提高机器学习在数据集上的表现。与单一模型相比，可以产生更好的性能。

为什么模型表现不同？模型假设不同，优化技巧不同，参数初始化不同。

集成学习两个重要概念：准确性和多样性($Diversity$)。准确性指的是个体学习器不能太差，要有一定的准确度。多样性则是个体学习器之间的输出要具有差异性。

集成学习分为两类。

- 第一种是并行集成学习，比如说装袋($Bagging$)和随机森林($Random\ Forest$)。利用模型之间的独立性对最终结果做加权预测。
- 另一种是串行集成学习，如提升($Boosting$)。通过权衡前面模型错误标记的数据来提升整体表现。

集成学习潜在的思想是即便某一个弱分类器得到了错误的预测，其他的弱分类器也可以将错误纠正回来，而且集成学习在各个规模的数据及上都有很好的策略。

- 数据集大：划分为多个小数据集，学习多个模型进行组合。
- 数据集小：利用 $Bootstrap$ 方法进行抽样，得到多个数据集，分别训练多个模型再进行组合。

2. 投票($Voting$)

$Voting$: 获得赞同越多的结果越有可能是真实的结果。

对于分类问题，有三种方式：

- 绝对多数投票：统计所有分类器的分类结果，如果某个结果出现频次超过总预测次数的一半，则预测为该标记；如果没有，则拒绝该预测。
- 相对多数投票(简单多数投票)：统计所有分类器的分类结果，如果某个分类得票次数最多，则预测为该标签。
- 加权投票：每个分类器在最终的结果中占据不同的权重，分类结果倾向于表现更好的模型。

3. 平均($Averaging$)

另一种方式就是平均($Averaging$)，它在回归和分类问题上都有不错的表现，能够提高 AUC 或者降低均方误差。

- 简单平均：$H(x)=\frac{1}{T}\sum\limits_{i=1}^{T}h_i(x)$
- 加权平均：$H(x)=\frac{1}{T}\sum\limits_{i=1}^{T}w_ih_i(x)$

平均预测常常会降低过拟合。在类与类间，你想要理想的平缓的将其分离，而单一模型的预测在边界间可能会有一些粗糙。

4. 模型结合($Combination$)

学习器结合可以带来很多好处，比如从统计上提升泛化性能。从计算上降低陷入局部极小点的风险，从表示上扩大假设空间以更好的近似。$Stacking$ 就是典型的模型结合思想。

5. 装袋($Bagging$)

核心：多次采样。

装袋即引导聚集算法($Bootstrap\ Aggregation$)，这种方法通过构造一系列弱分类器，然后以一定的方式将他们组合成一个强分类器，可以有效降低结果方差，避免过拟合。

5.1. $Bootstrap$ 采样

$Bootstrap$ 就是从一个原始样本中进行有放回的重复采样，采样大小和原始样本的大小相同，采样次数根据计算量而定。

当我们不知道样本分布的时候，$Bootstrap$ 方法最有用。$Bootstrap$ 分布和样本分布相似，因此可以用前者来估计后者。

5.2. 算法思路

现有数据集 $data$ ： $\mathbb{Z}=\{(x_i,y_i)\}_{i=1,2,\cdots,N}$ 。

$Bootstrap\ sample\ set$: $\mathbb{Z}^{*b},\ b=1,\cdots,N$ ，即有 b 个数据集。

对 b 个数据集构建 b 个分类器 $classifer$: $f_b(x)$ 。

$'Bagging'\ estimate$:

$$\hat{f}_{bag}(x)=\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{\times b}(x)$$

即平均 b 个弱分类器的结果，但是平均为什么能让分类器变好呢？我们假设需要求的分布为：

$$\mathbb{E}_{\hat{p}}\hat{f}^*(x), \quad 'Ture'\ bagging\ estimator$$

根据相关理论， $\hat{f}_{bag}(x)$ 是 $\mathbb{E}_{\hat{p}}\hat{f}^*(x)$ 的门特卡罗估计($Monte\ Carlo\ Estimate$ 最好估计)。

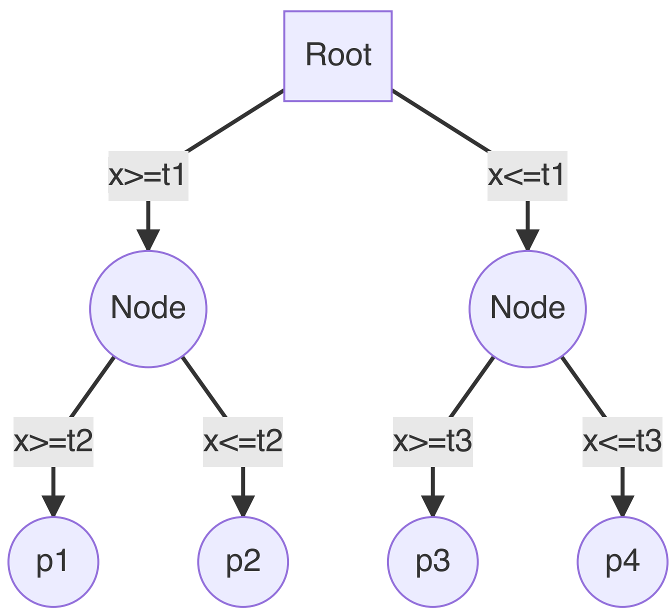
理论上说，任何弱分类器都可以通过装袋优化(树方法)，且能有效降低过拟合。但是强分类器不需要，因为本身不存在多样性，因此装袋没有意义。

5.3. $Bagging + Tree-based\ classifier$

将很多树合在一起：

$$\begin{matrix} 1.\left(\quad Tree \quad \right) & 2.\left(\quad Tree \quad \right) & \cdots & 100.\left(\quad Tree \quad \right) \\ \searrow & \downarrow & & \swarrow \\ & \sum & & \end{matrix}$$

$Tree$ 的结构：



5.4. 随机森林($Random\ Forest$)

其实就是 $Bootstrap + Averaging$ 。$random\ forest$ 有两种随机方式，第一种随机是指 $Bootstrap$ 采样得到不同的 $subset$ ，第二种随机是构造每棵树的时候，随机选一些特征来生成每棵决策树，而并非使用全部的特征去构造。

$$\begin{aligned} & For\ b = 1\ to\ B: \\ & \quad a)\ Draw\ a\ bootstrap\ sample\ \mathbb{Z}\ of\ size\ N. \\ & \quad b)\ Grow\ a\ random-forest\ tree\ T_b\ by\ resursively\ repeating. \\ & \qquad i).\ Select\ m\ features\ at\ random\ from\ P\ features. \\ & \qquad ii).\ Pick\ the\ best\ feature. \\ & \qquad iii).\ Split. \\ & \quad c)\ Bagging \end{aligned}$$

优点：能把所有的 $feature$ 对于分类的效果排序，即告诉哪个 $feature$ 最重要(
$n\ feature\ at\ from\ p\ features,\ best\ feature$)，该算法在特征过多时适用。(最优特征提取)

6. 提升($Boosting$)

串行学习，依次给样本赋权重，依次给分类器模型($M_1, M_2, \cdots$)赋权重。

核心：加权($weighting$)

6.1. 算法思路

现有数据集 $data$ ： $\mathbb{Z} = \{(x_i, y_i)\}_{i=1,2,\cdots,N}, \quad y_i \in \{-1, 1\}$ 。

给每个采样的样本赋权重， $data\ samples : w_i(data\ weights)$ 。

给每个分类器赋权重， $models : \alpha_m(model\ weights)$ 。

每个分类器模型的错误率， $error\ rate : \overline{err} = \frac{1}{N}\sum_{i=1}^{N} I(y_i \neq G(x_i))$ 。其中，I 是指标函数，如果括号里条件成立，则 $I = 1$ ，否则为 0 。

基于 $error\ rate$ ，衡量不同模型的差异，即加强重要的，拒绝不重要的。

6.2. $AdaBoost$ 训练中的分治法(分阶段优化)

$AdaBoost$ 的训练其实也采用了分治法的策略，主要利用了其中的分阶段优化法：即每次迭代时先训练弱分类器，然后确定弱分类器的权重参数。

$AdaBoost$ 算法在训练时的目标是最小化损失函数，假设强分类器为 $F(x)$ ，$x \in R^n$ 为训练集样本(特征向量)，$y = \pm 1$ 为标签值。单个训练样本的指数损失函数定义为：

$$L(y, F(x)) = \exp(-yF(x))$$

对于第 i 个弱分类器 $f_i(x)$，β_i 是弱分类器的权重，M 是弱分类器的总数。强分类器是弱分类器 $f_i(x)$ 的加权组合，定义为：

$$F(x) = \sum_{i=1}^{M} \beta_i f_i(x)$$

训练时依次训练每个弱分类器，将其依次加入强分类器中。将强分类器的计算公式带入到上面的损失函数中，得到训练第 j 个弱分类器时对整个训练集的损失函数为：

$$(\beta_j, f_j) = \arg\min_{\beta, f} \sum_{i=1}^{l} \exp(-y_i(F_{j-1}(x_i) + \beta f(x_i)))$$

这个式子有两个部分，第一部分是之前的迭代中，已经得到的强分类器 F_{j-1} ，第二部分是当前要训练的弱分类器 f 与其权重 β 的乘积对训练样本的损失函数。第一部分在之前的迭代中已经求出，可以看成常数，因此上式的目标函数可以简化为：

$$\min_{\beta, f} \sum_{i=1}^{l} w_i^{j-1} \exp(-\beta y_i f(x_i))$$

其中 $w_i^{j-1} = \exp(-y_i F_{j-1}(x_i))$ 定义为样本权重，它只和前面 $j-1$ 次迭代得到的强分类器有关，与当前的弱分类器、弱分类器的权重均无关。

利用分治法中的分阶段优化，这个目标函数可以分为两步求解。首先将 β 看成常数，求 f 的最优解。因为 y_i 和 $f(x_i)$ 的取值只能为 ± 1 ，且样本权重 β 非负，若想让上式最小化，必须让 $y_i = f(x_i)$ 。因此损失函数对于 $f(x)$ 的最优解为：

$$f_j = \arg\min_f \sum_{i=1}^{l} w_i^{j-1} I(y_i \neq f(x_i))$$

I 是指标函数，如果括号里条件成立，则 $I = 1$ ，否则为 0 。上式表示的最优解是使得对样本的加权误差最小的弱分类器。得到弱分类器后，再对 β 求优化问题，目标函数的优化目标可以表示成 β 的函数：

$$L(\beta) = \exp(-\beta) \times \sum_{y_i = f_j(x_i)} w_i^{j-1} + \exp(\beta) \times \sum_{y_i \neq f_j(x_i)} w_i^{j-1}$$

上式的前半部分是被当前弱分类器正确分类的样本，此时 $y_i f(x_i) = 1$ ，$\exp(-\beta y_i f(x_i)) = \exp(-\beta)$ 。后半部分是被当前弱分类器错误分类的样本，此时 $y_i f(x_i) = -1$ ，$\exp(-\beta y_i f(x_i)) = \exp(\beta)$ 。目标函数进一步可写成：

$$L(\beta) = (\exp(\beta) - \exp(-\beta)) \times \sum_{i=1}^{l} w_i^{j-1} I(y_i \neq f_j(x_i)) + \exp(-\beta) \times \sum_{i=1}^{l} w_i^{j-1}$$

推导过程如下：

$$\begin{aligned}
&\exp(-\beta) \cdot \sum_{y_i = f_j(x_i)} w_i^{j-1} + \exp(\beta) \cdot \sum_{y_i \neq f_j(x_i)} w_i^{j-1} \\
&= \exp(-\beta) \cdot \sum_{y_i = f_j(x_i)} w_i^{j-1} + \exp(-\beta) \cdot \sum_{y_i \neq f_j(x_i)} w_i^{j-1} - \exp(-\beta) \cdot \sum_{y_i \neq f_j(x_i)} w_i^{j-1} + \exp(\beta) \cdot \sum_{y_i \neq f_j(x_i)} w_i^{j-1} \\
&= \exp(-\beta) \cdot \sum_{i=1}^{l} w_i^{j-1} + (\exp(\beta) - \exp(-\beta)) \cdot \sum_{y_i \neq f_j(x_i)} w_i^{j-1} \\
&= \exp(-\beta) \cdot \sum_{i=1}^{l} w_i^{j-1} + (\exp(\beta) - \exp(-\beta)) \cdot \sum_{i=1}^{l} w_i^{j-1} I(y_i \neq f_j(x_i))
\end{aligned}$$

对 $L(\beta)$ 求导并令其为 0 ：

$$(e^{\beta} + e^{-\beta}) \times \sum_{i=1}^{l} w_i^{j-1} I(y_i \neq f_j(x_i)) - e^{-\beta} \times \sum_{i=1}^{l} w_i^{j-1}$$

该式两边同时除以 $\sum_{i=1}^{l} w_i^{j-1}$ ，得到关于 β 的方程：

$$(e^{\beta} + e^{-\beta}) \cdot err_j - e^{-\beta} = 0$$

最后得到：

$$\beta = \frac{1}{2} \ln \frac{1 - err_j}{err_j}$$

其中 err_j 为当前弱分类器对于训练样本集的加权错误率：

$$err_j = \frac{\sum\limits_{i=1}^{l} w_i^{j-1} I(y_i \neq f_j(x_i))}{\sum\limits_{i=1}^{l} w_i^{j-1}}$$

得到弱分类器及其权重之后，对强分类器进行更新：

$$F_j(x) = F_{j-1}(x) + \beta_j f_j(x)$$

下次迭代时的样本权重为：

$$w_i^j = w_i^{j-1} \cdot \exp(-\beta_j y_i f_j(x_i)) = w_i^{j-1} \exp\left[\beta_j I(y_i \neq f_j(x_i))\right]$$

6.3. $< AdaBoost\ M_1 >$ **算法流程**

假设最终输出为一个强分类器 $F(x)$ ，现有 l 个训练样本，M 个弱分类器，第 j 个弱分类器表示为 $f_j(x)$，β_j 是弱分类器的权重，w_i^{j-1} 为样本权重， err_j 为当前弱分类器对于训练样本集的加权错误率。

根据上面的公式推导，$AdaBoost$ 的算法流程如下：

$$\begin{aligned}
&1.\ Initialize: w_i^0 = \frac{1}{l}(i = 1, \cdots, N) \\
&2.\ For\ j = 1\ to\ M: \\
&\quad a)\ Fit\ f_j(x)\ to\ training\ data\ using\ w_i^{j-1} \\
&\qquad\quad f_j = \arg\min_f \sum_{i=1}^{l} w_i^{j-1} I(y_i \neq f(x_i)) \\
&\quad b)\ Compute: \\
&\qquad\quad err_j = \frac{\sum\limits_{i=1}^{l} w_i I(y_i \neq f_j(x_i))}{\sum\limits_{i=1}^{l} w_i^{j-1}} \\
&\quad c)\ \beta_j = \frac{1}{2}\ln\frac{1 - err_j}{err_j} \\
&\quad d)\ Update:\ w_i^j = w_i^{j-1} \exp\left[\beta_j I(y_i \neq f_j(x_i))\right] \\
&3.\ Output\ F(x) = sign[\sum_{i=1}^{M} \beta_i f_i(x)]
\end{aligned}$$

算法的流程图可表示为：

$$\begin{array}{cc}
w_i^{j-1} & \nwarrow \\
\downarrow & \uparrow \\
err_j & \uparrow \\
\downarrow & \uparrow \\
\beta_j \cdot model & \nearrow
\end{array}$$

6.4. $Gradient\ Boosting\ \&\ XGBoost$

自行翻阅相关资料...

6.5. 最佳实践(方差与偏差)

$Boosting$ 将弱分类器变为强分类器，但降低 $variance$ 方差的效果不明显(仍有过拟合)，主要是降低 $bias$ 偏差(欠拟合)。前文提到，降低方差使用 $Bagging$ 更合适。

可以给强分类器增加扰动，提高其分类效果。

$Bagging$ 不能很好地降低偏差 $bias$ ，因为偏差只和 $base\ learner$ 本身的能力有关；但能降低方差 $variance$ ，因为算法的实现需要重复采样。

7. 堆叠($Stacking$)

核心：模型融合。

主要思想是，首先使用标签和数据，用多个强分类器进行训练。之后对强分类器的输出概率特征图进行拼接，作为弱分类器的输入特征向量，再次使用标签和强分类器的输出进行训练，缓解过拟合问题。

可参考个人论文：GDN: A Stacking Network Used for Skin Cancer Diagnosis

8. 对比和总结

$Bagging$ 装袋的思路是使用 $Bootstrap$ 策略，每次采样整个数据集中的部分数据集，得到同一种分类器的不同参数的模型，最终分类结果通过投票，回归结果通过取均值得到。

区别：同一数据集的不同部分($different\ batch$)，同种模型的不同参数，分类结果由投票或平均得到。常用的有随机森林。

$Boosting$ 提升的思路是是每次根据分类器的表现进行调整，分类错的样本就改变权重。每次迭代循环，训练弱分类器，然后根据表现更新其权重。最终将这几个分类器进行加权求和。

区别：同一数据集，同种模型的不同参数，分类器由模型加权组合得到。常用的有 $AdaBoost, GradBoost, XGBoost$ 。

$Stacking$ 堆叠的思路是对整个数据集进行训练，但是采用不同种的分类器模型进行分级训练两次，以强分类器的输出作为弱分类器的输入。最后将强分类器的输出结果按第二级弱分类器进行加权。

区别：同一数据集，不同种模型，结果由不同模型输出多次堆叠得到。