机器学习讲义 (L18-B) 随机森林与集成学习

授课教师：王贝伦 / 助教：张嘉琦，黄旭，谈笑，徐浩卿

在前一章中，我们介绍了集成学习的基本概念，以及集成学习的聚合算法。聚合算法通过自助抽样得到多个训练集，将每个训练集单独训练出来的学习器通过投票法等集成方式聚合成一个大模型，以达到减小方差和防止过拟合的目的。聚合的决策树模型因决策树之间的相关性较大，对性能的提升不是很大。随机森林正是用来解决这个问题。

1 随机森林 (Random Forest)

随机森林分类器是聚合算法的拓展，它可以用来降低决策树之间的相关性。假设有 N 个样本，每个样本有 p 个特征，同聚合算法一样，需先自助抽样出 B 个大小不变的新训练集，再使用这 B 个训练集分别训练 B 棵决策树。与训练普通决策树不同的是，随机森林在分裂每一个节点时，会从 p 个特征中随机挑选出 $m(m < p)$ 个特征，再从这 m 个特征中选择使信息增益最大的特征作为该节点的分裂属性，剩下的 $p - m$ 个特征不予考虑。对每棵树重复以上操作，直到不能分裂。将这些决策树通过投票法聚合，这样就构成了随机森林。随机森林通过去除决策树相关性，可以被视为对聚合算法的改进。下面是 Breiman 提供的选择 m 的准则，但特殊问题仍要特殊对待，小心选择。

- 回归问题：$m = \frac{p}{3}$ 或 $m = \log_2 p$
- 分类问题：$m = \sqrt{p}$

决策森林通过去除相关性来提升聚合算法的效果，那为何相关性大的决策树效果不理想呢？下面我们来定量分析一下。假设每棵决策树的方差为 σ^2，那么如果决策树满足独立同分布的话，不难得到聚合后的方差为 $\frac{\sigma^2}{B}$。当 B 很大时，每个决策树平均下来的方差会很小。而如果决策树之间仅仅满足同分布的条件，两两之间的相关系数为 ρ 的话，那么计算平均下来的方差为

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

可以看出，当 $B \to \infty$ 时，方差的第二项趋近于 0，而第一项因为相关系数而始终影响方差的大小。正因如此，随机森林可以通过减小分裂时实际考虑的特征数 m，来减小树与树之间的相关性，因此方差也就减小了。

Algorithm 1: 随机森林算法

```
输入：样本集 Z，样本个数 N，决策树个数 B;
for b ← 1 to B do
    从样本集 Z 中自助抽样出新的大小为 N 的样本集 Z*;
    while 可以继续分裂 do
        从 p 个特征中随机挑选出 m 个特征;
        从 m 个特征中选择出分裂的最佳特征;
        分裂出两个个子节点;
    end
end
输出：决策树的集成 {T_b}_1^B;
```

2 堆叠 (Stacking)

堆叠，即 stacking 算法，也是集成学习的一种方法。如图1所示，它的基本思想是使用训练数据训练多个基学习器，然后使用基学习器的预测输出作为元学习器的输入并训练元学习器，以元学习器的预测结果作为最终预测结果。

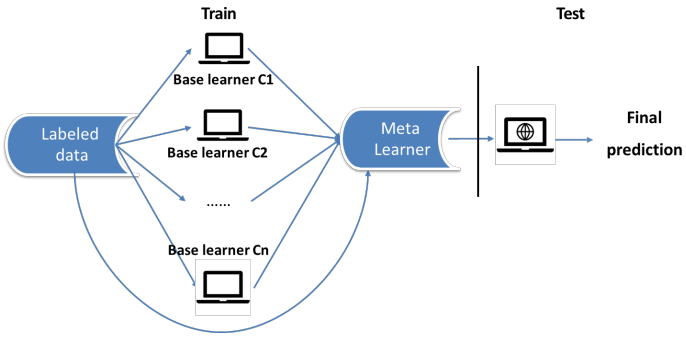

Figure 1: stacking 算法的基本思想

基学习器需要有较高的运算效率，较高的准确性和多样性。可以通过采样训练集，采样特征或使用不同的学习模型来作为基学习器。而元学习器也有多种选择，可以用非监督的投票法，加权平均等方法，也可以用监督学习的分类器。通常使用逻辑回归作为元学习器。下面举一个 stacking 算法的简单例子。如图2所示，首先将训练集分成两个子集，使用子集 1 来训练三个基学习器。

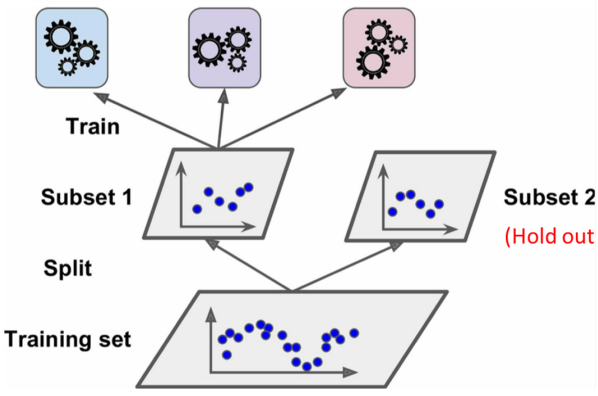

Figure 2: 训练基学习器

随后，将子集 2 输入三个已训练的基学习器，得到三组预测结果。将不同基学习器的预测结果作为新的特征，用来训练元学习器，最终得到一个集成的模型。

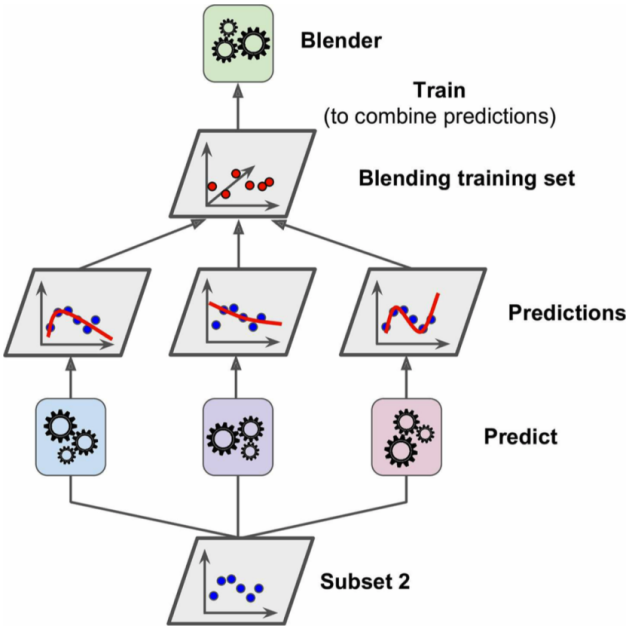

Figure 3: 训练元学习器

3 集成学习的提升方法（Boosting）

我们之前介绍过集成学习的目的是综合多个机器学习模型的结果，从而得到更好地预测结果。我们介绍过 bagging 方法，这里我们介绍另外一种简单的方法，提升方法（Boosting）。之前的 bagging 方法的特点就是公平：每个基础模型有相同的权重，即每个基础模型都有一票的投票权。得票多的结果作为集成学习的预测结果。而 boosting 方法却不同。具体来说，boosting 依次训练基础分类器，每次训练完成后会更新训练数据的权重。那些被分类错误的数据会拥有更大的权重。这样下一个基础训练器就会更加“关注”前一个模型无法处理或者预测错误的数据。一步步训练下去，模型会逐渐关注于那些难预测的样本，从而达到更好地预测效果。

我们下面用一个例子来简单说明 boosting 方法的流程。图4展示了一些苹果图像的样本分布，这些图像里面有些事真实苹果的图片，而另外一些则是苹果形状的标识。可以看到横轴的数据表示图像的颜色，纵轴的数据表示图像的亮度。我们现在要用 boosting 方法训练出一个能够分类这两类苹果的模型。

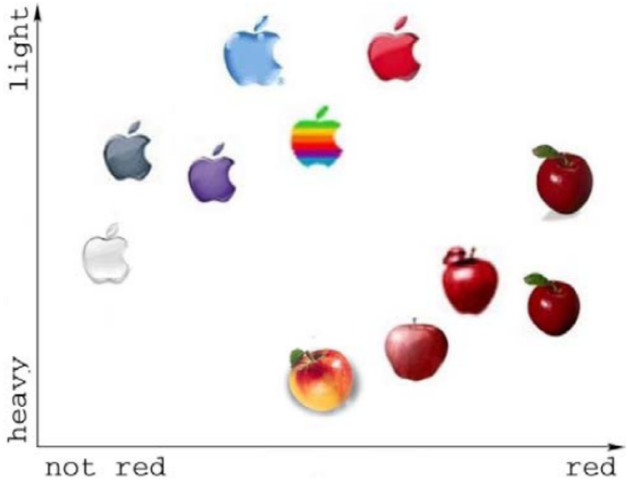

Figure 4: 苹果样本数据。

在第一步，我们认为所有样本的权重是相同的，并且训练出一个简单的模型，如图5所示，这个模型只是简单地用颜色来分类。可以看到有两个被圈出的样本的分类结果是错误的。

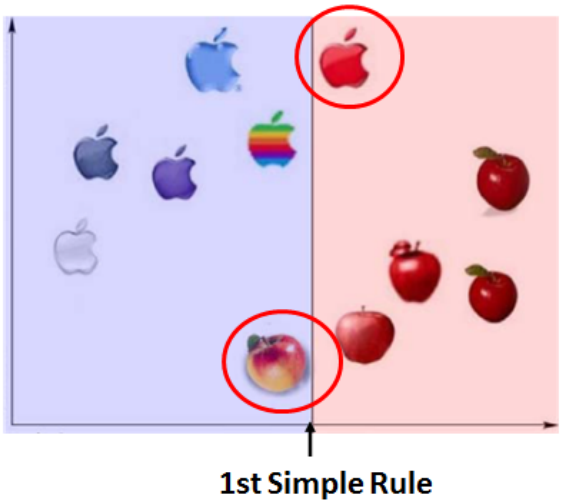

Figure 5: 第一步训练出的简单模型。

所以在第二步训练的模型中，我们更加关注这些分类错误的数据。得到的第二个简单模型如图6所示。可以看到之前错误分类的样本被分到正确的类别里面，然而又有新的错误分类样本出现，因此我们要再调整训练数据的权重继续进行训练。

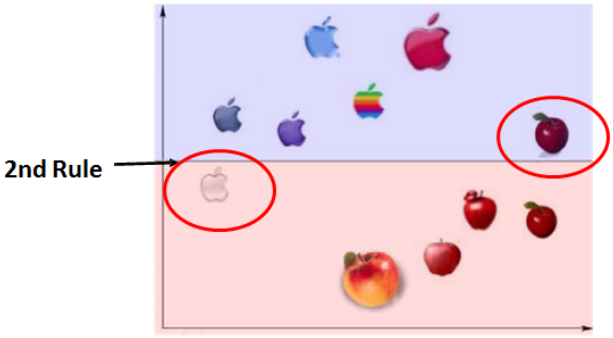

Figure 6: 第二步训练出的简单模型。

假设我们又进行了两步训练，训练得到了两个模型，如图7以及8所示。

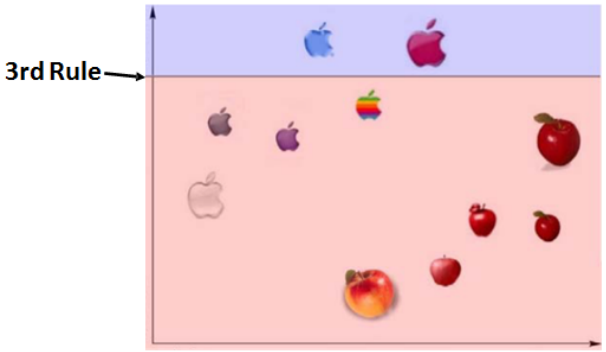

Figure 7: 第三步训练出的简单模型。

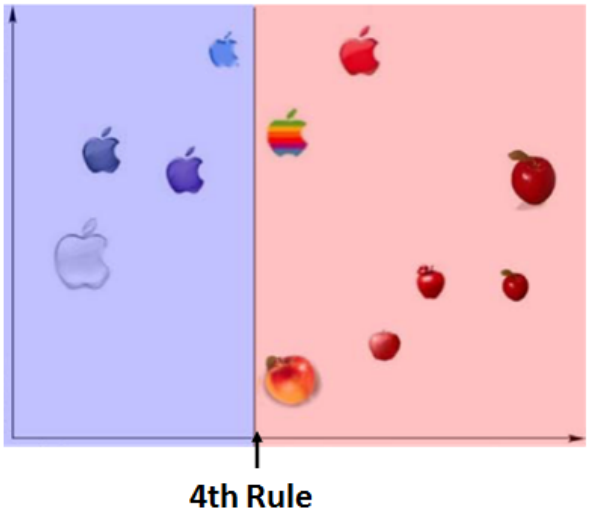

Figure 8: 第四步训练出的简单模型。

最后我们综合四个基础模型的结果（图9），得到一个完美将两类数据分类的分类器（图10）。

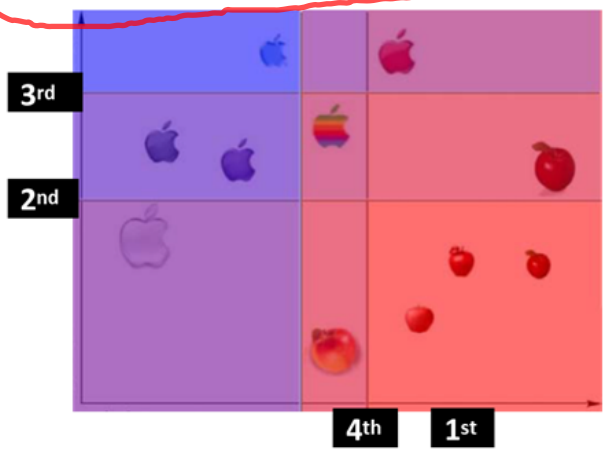

Figure 9: 整合四个基础模型。

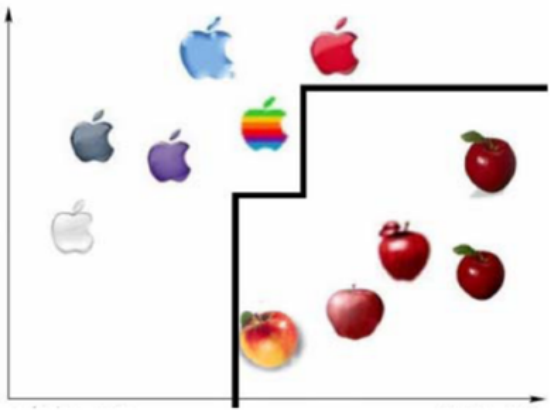

Figure 10: Boosting 方法得到的集成模型。

4 Adaboost 算法

Adaptive Boosting，即 Adaboost 是 Yoav Freund 和 Robert Schapire 提出的一种使用了 boosting 方法的集成学习模型 [4]。具体来说，假设我们有一个训练集 $\mathcal{D} = \{(x_i, y_i) \mid x_i \in \mathbb{R}^p, y_i \in \{0, 1\}, i = 1, 2, \cdots, n\}$，在 Adaboost 中我们训练第一个基础模型时假设所有数据的权重都相同，为 $w_i = \frac{1}{n}, i = 1, 2, \cdots, n$。然后我们把通过如下的迭代过程来完成 Adaboost 的训练：

1. 对于第 t 步，我们用训练集训练一个模型 h_t 来获得最小的分类误差
$$h_t(\mathcal{D}, w) = \text{argmin } \varepsilon_t \tag{2}$$
其中 $\varepsilon_t = \sum_{i=1}^{n} w_i * \mathbf{1}(y_i \neq \widehat{y}_i o)$ 表示所有分类错误的样本的权重之和，$\widehat{y}_i$ 是第 t 步的模型对于样本 x_i 的预测值。
2. 我们同时为这个第 t 步得到的模型分配一个权重 $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$.
3. 更新每个样本的权重 $w_i = w_i * \exp(-\alpha_i y_i \widehat{y}_i)$
4. 令 $t = t + 1$，回到 1 训练下一步的基础模型。

当我们训练出 T 个模型之后（T 一般是我们预先设定的需要训练的基础模型的数量），我们综合这些模型得到对于样本 x_i 的最终分类结果为
$$\widehat{y}_i = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x_i)) \tag{3}$$
其中 $h_t(x_i)$ 是不同基础模型对于样本 x_i 的分类结果。

5 XGBoost

XGBoost 是一个在许多平台和编程语言上发布的基于树结构的 boosting 方法的软件库。在近几年各类机器学习竞赛中，XGBoost 都获得非常优异的成绩。XGBoost 是基于树结构通过梯度方法实现的一种 boosting 方法。综合多个树结构的结果，XGBoost 可以获得更好地预测结果（图11）。这里我们呢只是简单介绍 XGBoost，详细的解释与 XGBoost 特性说明可以参考 XGBoost 的官方网站 [5] 以及官方文档 [6]。

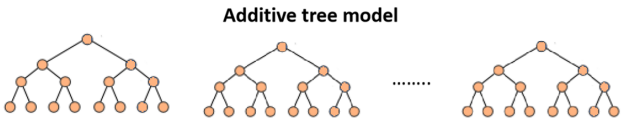

Figure 11: XGBoost 的树结构。

6 Bagging vs. Boosting

我们简单比较一下目前介绍的两种集成学习方法。

- Bagging 和 boosting 都集成了多个模型的结果，因此这两种方法都降低了预测结果的方差（variance）。
- Boosting 的一个优点在于它在每一步都常识得到最好的模型，所以模型会逐渐变得越来越复杂。因此 boosting 方法可以降低偏差（bias）。
- Boosting 和 bagging 的一个主要区别在于对待训练数据的不同。Bagging 的特征就是公平，每个基础模型是平等的，都拥有一票投票权。最终结果取决于投票结果。而 boosting 对于模型和数据都做了区别对待。

- 因为 boosting 需要顺序地预测不同的模型，因此 boosting 方法不容易并行实现。而 bagging 方法中每个模型时独立的，因此 bagging 更适合可以并行运行的环境。

引用

[1] Boosting Algorithms: AdaBoost, Gradient Boosting and XGBoost.

[2] Understanding AdaBoost: https://towardsdatascience.com/understanding-adaboost-2f94f22d5bfe

[3] XGBoost: https://en.wikipedia.org/wiki/XGBoost

[4] Freund, Yoav, Robert Schapire, and Naoki Abe. "A short introduction to boosting." Journal-Japanese Society For Artificial Intelligence 14.771-780 (1999): 1612.

[5] XGBoost: https://xgboost.ai/

[6] XGBoost Documentation: https://xgboost.readthedocs.io/en/latest//index.html

[7] Random Forest: https://en.wikipedia.org/wiki/Random_forest