关于 Boosting 算法在分类问题中的比较与应用

许卓岩

指导教师：邓爱姣　副教授

武汉大学数学与统计学院

2019 年 5 月 25 日

Overview

1. 概述
2. Boosting 与可加模型
   - Adaboost
   - 可加模型
   - GBM: 一种函数估计
3. XGBoost
   - 正则化下的目标函数
   - Newton-boosting
   - 树构建算法
4. 统计模拟

机器学习与集成学习

- 机器学习：机器学习是一种计算机系统改善自己在特定任务上统计算法和统计模型的研究。
- 集成学习：集成学习为机器学习中解决分类问题的一类标志性算法，它通过构建多个不同的分类模型（基分类器或弱分类器），再将这些分类器进行某种组合而得到强分类器进行预测。集成学习通过这种组合，可以获得比单一分类器更加稳定和优良的输出结果。

Boosting

- 根据个体学习器的生成方式和依赖关系，集成学习的算法可分为两大类：1. 个体学习器间存在强依赖关系，学习器串联生成的方法，如 Boosting，2. 个体学习器间不存在强依赖关系，可同时生成的方法，如 bagging 和随机森林。
- 从统计的角度来看，boosting 可以看成一种非参数学习算法，此方法不依赖于数据的特征或分布。随着现今数据种类的多样化，数据的分布或特性往往杂乱而不可知，这就加大了基于数据分布的参数推断或参数估计的难度，也提高了 boosting 这类非参算法的有效性。

Overview

Adaboost

假设我们有 n 个训练样本 $(x_1, y_1), \ldots, (x_n, y_n)$，其中 x_i 是表示一系列特征（属性）的向量，y_i 是每个样本的标签，取值为 1 或-1. 我们定义

$$F(x) = \sum_{i=1}^{M} c_m g_m(x)$$

其中 $g_m(x)$ 指每一次计算的弱分类器，其输入每一个样本的属性可预测其标签，预测结果只比随即猜测稍好一些 $(error \leq 50\%)$，其中 c_m 是每一个弱分类器所占的权重。

Adaboost

Algorithm 1 *Adaboost.M1.*

1: Initialize the observation weights $w_i = 1/N, i = 1, \ldots, N$

2: **for** $m = 1, 2, \ldots, M$ **do**

3: (a) Fit the weak classifier $g_m(x) \in \{-1, 1\}$ on the training data with weights w_i.

4: (b) Compute error:

$$err_m = E_w[1_{(y \neq g_m(x))}] = \frac{\sum_{i=1}^{N} w_i 1(y_i \neq g_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

5: (c) Compute $c_m = log((1 - err_m)/err_m)$.

6: (d) Set $w_i \leftarrow w_i exp[c_m 1(y_i \neq g_m(x_i))], i = 1, 2, \ldots, N$, renormalize weights so that $\sum_i w_i = 1$.

7: Output the classifier $sign[\sum_{i=1}^{M} c_m g_m(x)]$

图: Adaboost algorithm

可加模型

在统计中，可加模型是一种非参回归方法，可加模型将多种基函数或简单光滑器结合，组成一个复杂的回归模型，达到预测的目的。在 Boosting 方法中，主要用到的是 Forward stagewise，在每次添加基函数或称弱分类器（前面所说的新变量）时不修改之前基函数的形式与系数：

Algorithm 2 *Forward stagewise additive modeling*

1: Initialize $F_0(x) = 0$.
2: **for** $m = 1, 2, \ldots, M$ **do**
3: (a) Compute

$$(\beta_m, a_m) = \arg\min_{\beta,a} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \beta h(x_i, a))$$

4: (b) Set$F_m(x) = F_{m-1}(x) + \beta_m h(x, a_m)$

图: 可加模型

指数损失与 Adaboost

指数损失:$L(y, F(x)) = e^{-yF(x)}$ 对于 Adaboost 来说，基函数或弱分类器用 $g_m(x) \in \{-1, 1\}$ 指代:

$$(\beta_m, g_m) = \arg\min_{\beta, g} \sum_{i=1}^{N} \exp\{-y_i(F_{m-1} + \beta g(x_i))\}$$

由于在 forward stagewise 中，第 m 步添加的分类器和系数 β_m 对前面不造成改变:

$$(\beta_m, g_m) = \arg\min_{\beta, g} \sum_{i=1}^{N} w_i^{(m)} \exp\{-\beta y_i g(x_i)\} \tag{1}$$

其中 $w_i^{(m)} = \exp(-y_i F_{m-1}(x_i))$. 由于每一步的 $w_i^{(m)}$ 与 β 与 $g(x)$ 都无关，其可被看成作用于每一个样本上的权重。

函数估计

函数估计指一种在函数空间上而不是参数空间上的数值优化。GBM 与 Adaboost 的主要区别在于损失函数的不同，且由于直接优化目标函数的困难性，GBM 使用梯度下降法作为算法核心。
在函数估计中，我们的最终目的是寻找优化函数：

$$F = \arg\min_F E_{y,x} L(y, F(x)) = \arg\min_F E_x[E_y L(y, F(x))|x]$$

这是一种函数空间上的非参方法，在实际应用中，当 (y,x) 的联合分布的信息蕴含在有限的样本 $\{y_i, x_i\}_1^N$ 中时：

$$F = \arg\min_F \sum_{i=1}^{N} L(y_i, F(x_i)) \tag{2}$$

GBM 中的梯度下降

我们在 m 步需要解决:

$$(\beta_m, a_m) = \arg\min_{\beta,a} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \beta h(x_i, a)) \tag{3}$$

直接解出往往比较困难，我们在这里借用梯度下降的思想，每一次迭代向梯度方向跨出一步。
具体做法为在每一步引入一个弱分类器，使其尽可能去拟合这一步得出的梯度。即在现有数据上，使得每一步的弱分类器与当前的梯度（函数前进方向）尽可能“相近（或高度相关)”，在“相近”的定义上，我们运用了平方损失法则。

GBM

Algorithm 3 *Gradient Boosting*

1: Initialize $F_0(x) = \arg\min_\beta \sum_{i=1}^N L(y_i, \beta)$.
2: **for** $m = 1, 2, \ldots, M$ **do**
3: (a)

$$\tilde{y}_i = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x_i)=F_{m-1}(x_i))}, i = 1, \ldots, N$$

4: (b) $a_m = \arg\min_{a,\beta} \sum_{i=1}^N [\tilde{y}_i - \beta h(x_i, a)]^2$
5: (c) $\beta_m = \arg\min_{\beta,a} \sum_{i=1}^N L(y_i, F_{m-1}(x_i) + \beta h(x_i, a_m))$
6: (d) $F_m(x) = F_{m-1}(x) + \beta_m h(x, a_m)$

图: Gradient Bosting algorithm

Overview

1. 概述
2. Boosting 与可加模型
   - Adaboost
   - 可加模型
   - GBM: 一种函数估计
3. XGBoost
   - 正则化下的目标函数
   - Newton-boosting
   - 树构建算法
4. 统计模拟

概述

XGBoost 是在 boosting 的基础上进一步发展得到。XGBoost 最成功的因素就是它在很多场景的可延展性，其更好的运用了二次梯度来优化函数，此思想被 Friedman[*a statistical view of boosting,1999*] 首次提出，XGBoost 在正则化条件下，对二次梯度有着更好的应用。

正则化下的目标函数

假设数据集有 n 个样本，p 个特征，我们在这里，将基分类器（回归树）的一般形式写为：

$$f(x) = \sum_{j=1}^{J} \gamma_j 1(x \in R_j) = w_{q(x)}$$

$$q : \mathbb{R}^p \longrightarrow J,\ w \in \mathbb{R}^J, w_j = \gamma_j$$

其中 q 代表把样本映射到叶编号的每一棵树的结构，每一棵树的深度相同，叶数量也相同。J 代表每一棵树叶的数量，即最后划分区域的数量。每一个 f_m 都有着其唯一的树结构和叶的预测值 w(或称 η)。与决策树不同，每一个回归树的叶预测值都为连续值，我们用 w_i(即 η_i) 代表第 i 个区域的预测值 (其中 $i = 1, \ldots, J$)。对于每一个需要预测的个体，我们将其在每一棵树对应的叶预测值 w 相加，得到最终的预测值。

正则化下的目标函数

将正则化项直接加入优化目标函数中:

$$\mathcal{L}(\phi) = \sum_i l(y_i, F(x_i)) + \sum_m \Omega(f_m)$$

$$\text{where } \Omega(f) = \gamma J + \frac{1}{2}\lambda \|w\|^2$$

其中 l 为可微的凸损失函数，$\Omega(f_m)$ 衡量了基分类器的复杂度，其中 γ, λ 为正则化中罚项的系数。

Newton-Boosting

用可加模型的角度取看待目标函数，用 $F_m(x)$ 表示在第 m 次迭代的值，则我们有:

$$\mathcal{L}^{(m)} = \sum_i l(y_i, F_{m-1}(x_i) + f_m(x_i)) + \sum_m \Omega(f_m)$$

我们需要找到每一步最适合的 f_m 来最好的拟合当前模型，最小化损失函数。

Newton-Boosting

在这里，我们引入二次梯度的思想，通过直接对 $l(y_i, F_{m-1}(x_i) + f_m(x_i))$ 在 $l(y_i, F_{m-1}(x_i))$ 附近做二阶泰勒展开，我们有：

$$l\left(y, F^{(m-1)}(x) + f_m(x)\right) \approx l\left(y, F^{(m-1)}(x)\right) + g_m(x) f_m(x) + \frac{1}{2} h_m(x) f_m(x)^2$$

其中：

$$g_m(x) = \left[\frac{\partial L(y, F)}{\partial F}\right]_{F=F_{m-1}(x)} \quad h_m(x) = \left[\frac{\partial^2 L(y, F)}{\partial F^2}\right]_{F=F_{m-1}(x)}$$

则优化函数可写为：

$$\mathcal{L}^{(m)} \simeq \sum_{i=1}^{n} \left[l(y_i, F_{m-1}) + g_{mi} f_m(x_i) + \frac{1}{2} h_{mi} f_m^2(x_i)\right] + \Omega(f_m)$$

去除常数项，带入 $\Omega(f_m)$ 的定义，我们有:

$$\tilde{\mathcal{L}}^{(m)} = \sum_{j=1}^{J} \left[\left(\sum_{i \in R_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in R_j} h_i + \lambda \right) w_j^2 \right] + \gamma J$$

对于一个固定的结构的树，我们只需找到最优的预测值 w_j^*:

$$w_j^* = -\frac{\sum_{i \in R_j} g_i}{\sum_{i \in I_j} h_i + \lambda}, \quad j = 1, \ldots, J$$

Newton-Boosting

经计算得目标函数最优值为:

$$\tilde{\mathcal{L}}^{(m)}(q) = -\frac{1}{2}\sum_{j=1}^{J}\frac{\left(\sum_{i\in R_j} g_i\right)^2}{\sum_{i\in R_j} h_i + \lambda} + \gamma J \tag{4}$$

式 [4] 可被当作一种用于衡量一个基函数是否优秀的得分函数。然而通常来说，遍历所有的树结构来找到最优解是不可能的。一个精确算法是从单节点树开始，不断添加 split 来构建树，看得分的变化，用 R_L, R_R 来表示一个区域被分开为两个区域后两个子区域的数据集，且 $R = R_L \cup R_R$, 则每一次的信息熵损失函数可被表示为

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in R_L} g_i\right)^2}{\sum_{i\in R_L} h_i + \lambda} + \frac{\left(\sum_{i\in R_R} g_i\right)^2}{\sum_{i\in R_R} h_i + \lambda} - \frac{\left(\sum_{i\in R} g_i\right)^2}{\sum_{i\in R} h_i + \lambda}\right] - \gamma \tag{5}$$

树构建算法

通过式 [5] 提出的熵增法则去构建树。精确的树构建算法为在所有特征上遍历所有可能的结点，此算法被称为精确贪婪算法：

Algorithm 7 *Exact Greedy Algorithm for Split Finding*

Input R, instance(obs) set of current node
Input d, dimension feature
gain $\longleftarrow 0, G \longleftarrow \sum_{i \in R} g_i, H \longleftarrow \sum_{i \in R} h_i$
for $k = 1, 2, \ldots, d$ **do**
    $G_L \longleftarrow 0, H_L \longleftarrow 0$
    for j *in sorted*$(R, by\ x_{jk})$ **do**
        $G_L \longleftarrow G_L + g_j, H_L \longleftarrow H_L + h_j$
        $G_R \longleftarrow G - G_L, H_R \longleftarrow H - H_L$
        $Score \longleftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}))$
Output:Split with max score

图：精确贪婪算法

1. 概述

2. Boosting 与可加模型
   - Adaboost
   - 可加模型
   - GBM: 一种函数估计

3. XGBoost
   - 正则化下的目标函数
   - Newton-boosting
   - 树构建算法

4. 统计模拟

MNIST

MNIST 是一个关于手写数字的大数据集，在图形处理和机器学习中都有着广泛的应用。MNIST 数据集一共有 60000 张训练图片，10000 张测试图片。每一张图片都被标准化为尺寸为 28 乘 28 的黑白图片。选取其中 5000 张来作为训练集与测试集。我们将每一张图片的像素值拉成一个向量，作为此样本的输入值，其长度为 $28 \times 28 = 756$。则每一个样本的预测变量（特征）为长度为 756 的向量，响应变量为 0 至 9 中的一位，即一共有 0 至 9 十类，每一类都有 500 张图片。取其中每一类的五分之一作为测试集，一共有 1000 张图片作为测试集，4000 张图片作为训练集。

默认参数

Adaboost	0.626
GBM(ls loss)	0.348
GBM(lad loss)	0.389
GBM(huber loss)	0.334
GBM(deviance)	0.912
XGboost	0.921

其中 Adaboost 与 GBM 的基分类器数目均为 100，分类器为决策树。可以看出，GBM 运用 deviance 作为损失函数，即 logistic GBM，时的效果明显好于其他损失函数，也优于分类的 Adaboost。XGboost 的效果在所有方法中最优。

交叉验证

分类器的交叉验证结果如下：

Adaboost	GBM(deviance)	XGboost
0.578	0.913	0.928
0.507	0.919	0.914
0.494	0.895	0.913
0.385	0.917	0.921
0.490	0.917	0.918

由此看出，第一次训练的结果具有一定代表性，交叉验证显示的精确度与之前的结果相符。XGBoost 仍然有最好的效果，而 GBM 稍次之，Adaboost 的结果并不理想。

参数调节

使用 Greedy search 的方法，找到对于每一个分类器最优的参数，如基分类器的个数，每一个决策树的深度，优化步长等等。将每一个参数带入分类器进行训练，选取结果最好的参数结果。为了方便比较，我们只选取分类器的结果进行比对。

Adaboost

Adaboost 使用最优参数 (47 个基分类器，优化步长 0.2 等) 得到的最好预测结果为 69.6%. 其中，混淆矩阵的 heatmap 为:

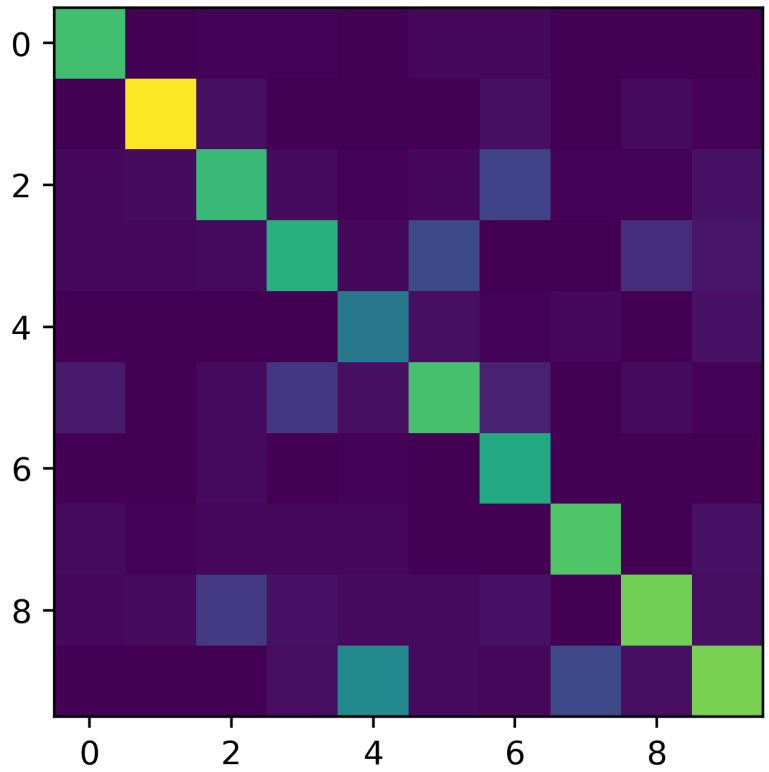
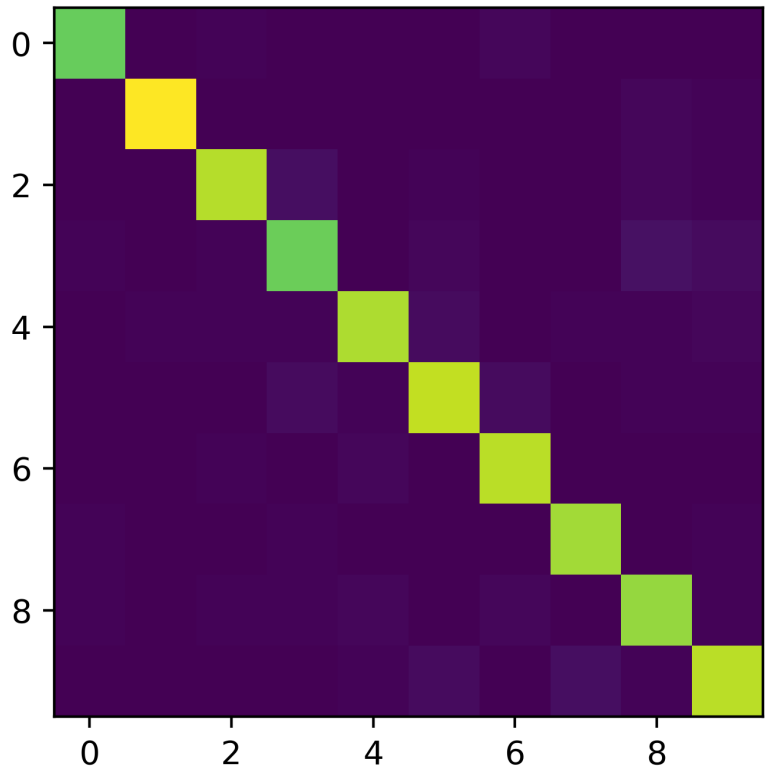


图: Adaboost heatmap

GBM

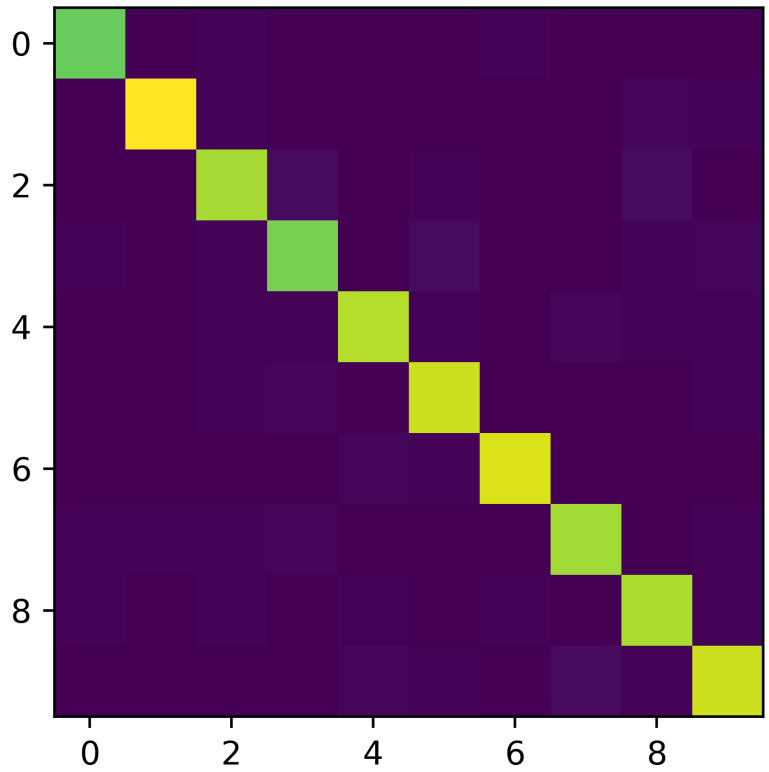
GBM 使用最优参数 (200 个基分类器，优化步长 0.2 等) 得到的最好预测结果为 93.2%. 其中，混淆矩阵的 heatmap 为：



图：GBM heatmap

XGBoost

XGBoost 使用最优参数 (200 个基分类器，优化步长 0.25 等) 得到的最好预测结果为 94.7%. 其中，混淆矩阵的 heatmap 为：



图：XGBoost heatmap

实验结论

由以上结果可知，Adaboost 的结果较好，在可接受范围。相比之下，GBM 与 XGBoost 有着更好的预测结果，也是它们现在被广泛使用的原因，当 GBM 的损失函数为 deviance 时，对于分类问题有着很高的预测精度。XGBoost 因使用二次梯度来优化函数，有着更高的收敛率。但实际收敛的时间并不占优，也许与计算二次梯度的计算复杂度有关。

References

- Friedman, Jerome, et al. "Special Invited Paper. Additive Logistic Regression: A Statistical View of Boosting." The Annals of Statistics, vol. 28, no. 2, 2000, pp. 337–374. JSTOR, www.jstor.org/stable/2674028.
- Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.
- Efron, Bradley, et al. "Least angle regression." The Annals of statistics 32.2 (2004): 407-499.
- Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning. Vol. 1. No. 10. New York: Springer series in statistics, 2001.
- Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016.
- Nielsen, Didrik. Tree Boosting With XGBoost-Why Does XGBoost Win" Every" Machine Learning Competition?. MS thesis. NTNU, 2016.

谢谢！