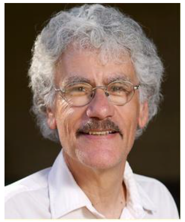
3.4 GBDT

CSDN学院

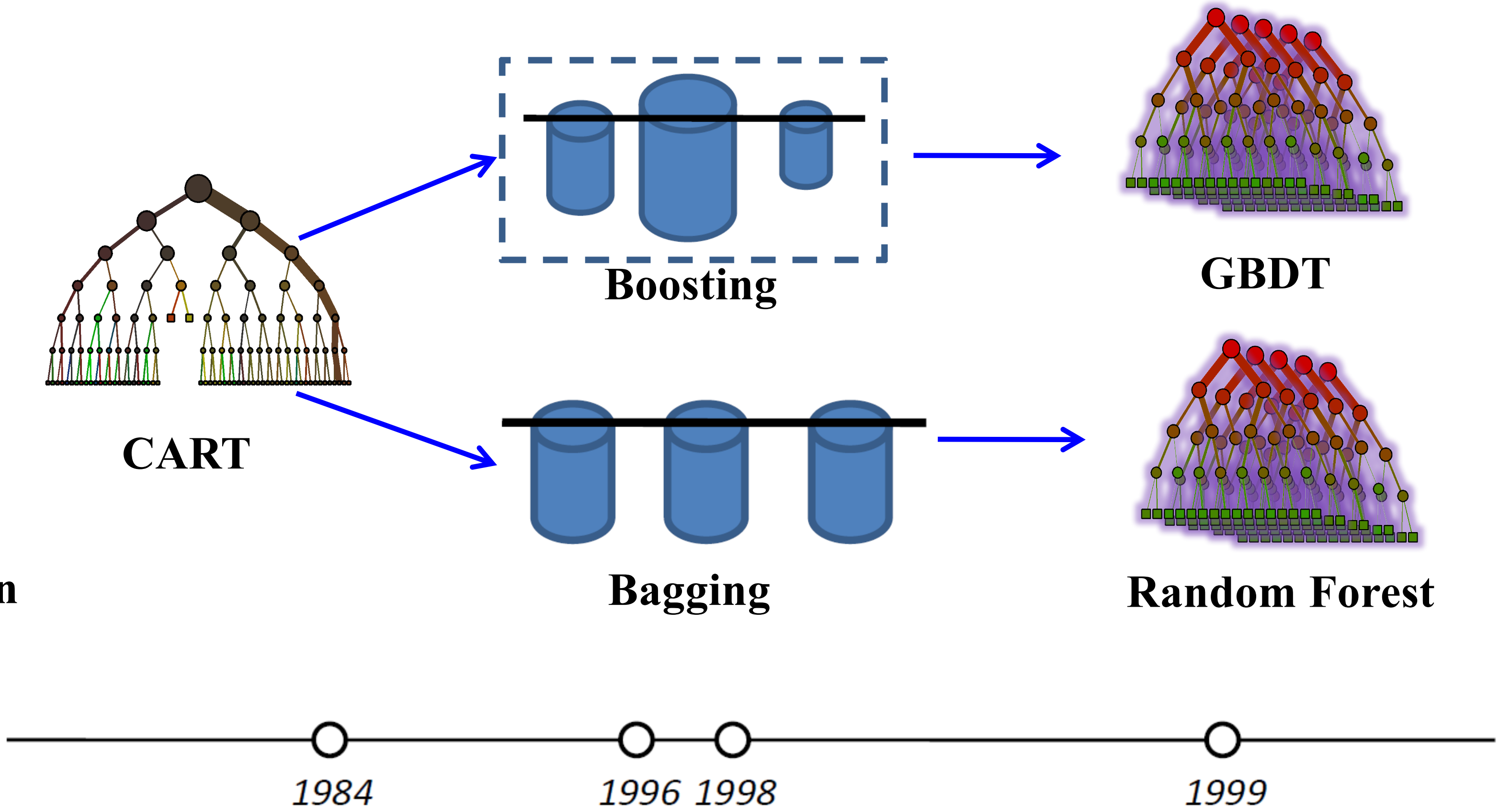
2017年11月

A brief history of CART



Jerome H. Friedman

Leo Breiman



GBDT (Gradient Boosting Decision Tree)

- GBDT在工业界应用广泛，通常被用于点击率预测，搜索排序等任务。
- GBDT也是各种数据挖掘竞赛的致命武器，据统计Kaggle上的比赛有一半以上的冠军方案都是基于GBDT。

▶大纲

CSDN
不 止 于 代 码

- Boosting
- AdaBoost
- Gradient Boosting
- GBM in Scikit learn

Boosting

- Boosting: 将弱学习器组合成强分类器
  - 构造一个性能很高的预测（强学习器）是一件很困难的事情
  - 但构造一个性能一般的预测（弱学习器）并不难
  - 弱学习器：性能比随机猜测好（层数不深的CART是一个好的选择）
- 亦可视为一种自适应基模型：

$$f(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m \phi_m(\mathbf{x})$$

  - 其中$\phi_m(\mathbf{x})$ 为基函数 / 弱学习器。

AdaBoost

CSDN 不 止 于 代 码

- 样本权重 / “过滤 “
  - 没有先验知识的情况下，初始的分布为等概分布，即训练集如果有N个样本，每个样本的分布概率为 $1/N$
  - 每次循环后提高误分样本的分布概率，误分样本在训练集中所占权重增大，使得下一次循环的弱学习器能够集中力量对这些误分样本进行判断

- 模型组合: 弱学习器线性组合
  - 准确率越高的弱学习机权重越高
  - $f(\mathbf{x}) = \mathrm{sgn}(\sum_{m=1}^{M} \alpha_m \phi_m(\mathbf{x}))$

AdaBoost M1算法

- 给定训练集：$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$, 其中. $y_i \in \{1, -1\}$ 表示$\mathbf{x}_i$的类别标签
- 训练集上样本的初始分布：$w_{1,i} = \frac{1}{N}$
- 对 $m= 1 : M$ ，
- 对训练样本采用权重$w_{m,i}$计算弱分类器$\phi_m(\mathbf{x})$
- 计算该弱分类器在分布 w_m上的误差：$\varepsilon_m = \frac{\sum_{i=1}^N w_{m,i}\ \mathbb{I}(\phi_m(\mathbf{x}_i) \neq \mathbf{y}_i)}{\sum_{i=1}^N w_{m,i}}$
- 计算该弱分类器的权重：$\alpha_m = \frac{1}{2} log \frac{1-\varepsilon_m}{\varepsilon_m}$
- 更新训练样本的分布: $w_{m+1,i} = \frac{w_{m,i} exp(-\alpha_m \mathbf{y}_i \phi_m(\mathbf{x}_i))}{Z_m}$

  其中Z_m 为归一化常数，使得 w_{m+1}是一个分布。
- 最后的强分类器为：$f(\mathbf{x}) = \text{sgn}(\sum_{m=1}^M \alpha_m \phi_m(\mathbf{x}))$

$\mathbb{I}(condition)$: 指示（ Indicator ）函数
满足条件值为1，否则为0

▶证明

$$f(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m \phi_m(\mathbf{x})$$

- 1. 对w_{M+1} 进行迭代展开

$$w_{M+1,i} = w_{M,i}\frac{exp\left(-\alpha_M y_i \phi_M(\mathbf{x}_i)\right)}{Z_M} = w_{1,i}\frac{exp\left(-y_i \boxed{\sum_{m=1}^{M} \alpha_m \phi_m(\mathbf{x}_i)}\right)}{\prod_{m=1}^{M} Z_m}$$

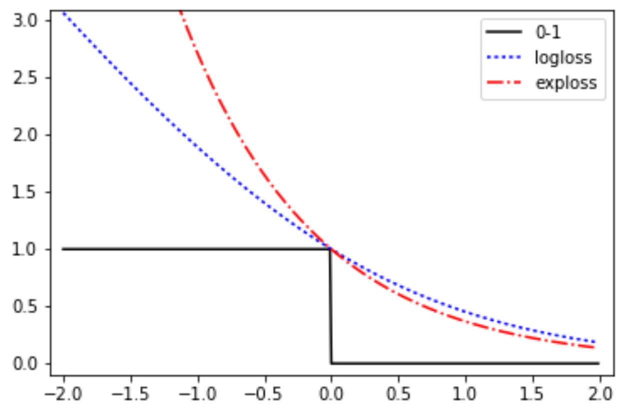
$$= w_{1,i}\frac{exp\left(-y_i f(\mathbf{x}_i)\right)}{\prod_{m=1}^{M} Z_m}$$

- 由于w_{M+1}是一个分布，所以$\sum_{i=1}^{N} w_{M+1,i} = 1$
- 所以$\prod_{m=1}^{M} Z_m = w_{1,i}\sum_{i=1}^{N} exp\left(-y_i f(\mathbf{x}_i)\right) = \frac{1}{N}\sum_{i=1}^{N} exp\left(-y_i f(\mathbf{x}_i)\right)$。

▶证明（cont.)

$$\prod_{m=1}^{M} Z_m = \frac{1}{N}\sum_{i=1}^{N} exp(-y_i f(\mathbf{x}_i))$$

- 2. 训练误差为：$ERR_{train}\big(f(\mathbf{x})\big) = \frac{1}{N}\left|\left\{i: y_i \neq sgn\big(f(\mathbf{x}_i)\big)\right\}\right|$



$$= \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1 & y_i \neq sgn\big(f(\mathbf{x}_i)\big) \\ 0 & esle\end{cases}$$ 0-1损失

$$\leq \frac{1}{N}\sum_{i=1}^{N}\exp\big(-y_i f(\mathbf{x}_i)\big)$$ 指数损失

$$= \prod_{m=1}^{M} Z_m$$

▶证明（cont.）

$$\prod_{m=1}^{M} Z_m = \frac{1}{N}\sum_{i=1}^{N} exp(-y_i f(\mathbf{x}_i))$$

$$Z_m = \sum_{i=1}^{N} w_{m,i} exp(-\alpha_m y_i \phi_m(\mathbf{x}_i))$$

- 3. 证明弱分类器权重为$\alpha_m = \frac{1}{2} log \frac{1-\varepsilon_m}{\varepsilon_m}$
- 问题：给定弱分类器的集合$\Delta = \{\phi_1(\mathbf{x}), \dots, \phi_M(\mathbf{x})\}$，确定弱分类器$\phi_m$及其权重$\alpha_m$
- $(\phi, \alpha)^* = \underset{\phi,\alpha}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N} exp\left(-y_i f(\mathbf{x}_i)\right) = \underset{\phi,\alpha}{\operatorname{argmin}} \prod_{m=1}^{M} Z_m$
- 具体实现时，首先选一个错误率最小的弱分类器ϕ_m，然后确定其权重α_m：
- $\frac{\partial Z_m}{\partial \alpha_m} = \frac{\partial \sum_{i=1}^{N} w_{m,i} exp\left(-\alpha_m y_i \phi_m(\mathbf{x}_i)\right)}{\partial \alpha_m}$
- $= -\sum_{i=1}^{N} w_{m,i} y_i \phi_m(\mathbf{x}_i) exp\left(-\alpha_m y_i \phi_m(\mathbf{x}_i)\right)$

$$\prod_{m=1}^{M} Z_m = \frac{1}{N} exp(-y_i f(\mathbf{x}_i))$$

$$Z_m = \sum_{i=1}^{N} w_{m,i} exp(-\alpha_m y_i \phi_m(\mathbf{x}_i))$$

- $\frac{\partial Z_m}{\partial \alpha_m} = -\sum_{i=1}^{N} w_{m,i} y_i \phi_m(\mathbf{x}_i) exp(-\alpha_m y_i \phi_m(\mathbf{x}_i))$
- $= \begin{cases} -\sum_{\mathbf{x}_i \in A} w_{m,i} exp(-\alpha_m) & if\ \mathbf{x}_i \in A, A = \{\mathbf{x}_i : y_i \phi_m(\mathbf{x}_i) = 1\} \\ \sum_{\mathbf{x}_i \in \bar{A}} w_{m,i} exp(\alpha_m) & if\ \mathbf{x}_i \in \bar{A}, \bar{A} = \{\mathbf{x}_i : y_i \phi_m(\mathbf{x}_i) = -1\} \end{cases}$ 分类正确的样本集合 / 分类错误的样本集合
- $\frac{\partial Z_m}{\partial \alpha_m} = 0 \ ==> \sum_{\mathbf{x}_i \in A} w_{m,i} exp(-\alpha_m) = \sum_{\mathbf{x}_i \in \bar{A}} w_{m,i} exp(\alpha_m)$ 两边同乘以$exp(\alpha_m)$
- $\underbrace{\sum_{\mathbf{x}_i \in A} w_{m,i}}_{1-\varepsilon_m} = exp(2\alpha_m) \underbrace{\sum_{\mathbf{x}_i \in \bar{A}} w_{m,i}}_{\varepsilon_m} \Rightarrow 1 - \varepsilon_m = \varepsilon_m exp(2\alpha_m)$

$$\alpha_m = \frac{1}{2} log \frac{1 - \varepsilon_m}{\varepsilon_m}$$

错误率小的弱分类器的权重更大

大纲

不止于代码

- Boosting
- Gradient Boosting
- XGBoost
- LightGBM

▶ 前向逐步递增

Forward stagewise additive modeling

- 还可以从另外一个角度来看AdaBoost：前向逐步递增
  - 要找到最优的f很难 → 每次递增

- 损失函数：$L(f(\mathbf{x}), y)$
- 目标函数：$\min_{f} \frac{1}{N} \sum_{i=1}^{N} L(f(\mathbf{x}_i), y_i)$
- 前向逐步递增
  - 初始化：$f_0(\mathbf{x}) = \operatorname{argmin}_{f} \frac{1}{N} \sum_{i=1}^{N} L(f(\mathbf{x}_i), y_i)$
  - 递增：$(\beta_m, \phi_m) = \operatorname{argmin}_{\beta,\phi} \frac{1}{N} \sum_{i=1}^{N} L(f_{m-1}(\mathbf{x}_i) + \beta\phi(\mathbf{x}_i), y_i)$

$$f_m(\mathbf{x}_i) = f_{m-1}(\mathbf{x}_i) + \beta_m \phi_m(\mathbf{x}_i)$$ 前向逐步递增

AdaBoost as前向逐步递增

- 将指数损失$L(f(\mathbf{x}), y) = exp(-yf(\mathbf{x}))$代入，
- 第m步，最小化
- $L_m = \sum_{i=1}^{\mathrm{N}} L(f(\mathbf{x}_i), y_i)$
- $= \sum_{i=1}^{\mathrm{N}} exp\left(-y_i\left(f_{m-1}(\mathbf{x}_i) + \beta\phi(\mathbf{x}_i)\right)\right)$
- $= \sum_{i=1}^{\mathrm{N}} \underbrace{exp\left(-y_i f_{m-1}(\mathbf{x}_i)\right)}_{w_{m,i}} exp\left(-y_i\beta\phi(\mathbf{x}_i)\right)$
- $= \left(e^{-\beta} \sum_{y_i=\phi(\mathbf{x}_i)} w_{m,i} + e^{\beta} \sum_{y_i\neq\phi(\mathbf{x}_i)} w_{m,i}\right)$
- $= \left(\left(e^{\beta} - e^{-\beta}\right) \sum_{i=1}^{N} w_{m,i}\, \mathbb{I}\left(y_i \neq \phi(\mathbf{x}_i)\right) + \left(e^{-\beta}\right) \sum_{i=1}^{N} w_{m,i}\right)$

$$\boxed{\begin{cases} y_i = \phi(\mathbf{x}_i) & y_i\phi(\mathbf{x}_i) = 1 \\ y_i \neq \phi(\mathbf{x}_i) & y_i\phi(\mathbf{x}_i) = -1 \end{cases}}$$

▶ AdaBoost as前向逐步递增(cont.)

CSDN 不止于代码

- $L_m = \left(\left(e^{\beta} - e^{-\beta}\right) \sum_{i=1}^{N} w_{m,i}\, \mathbb{I}\left(y_i \neq \phi(\mathbf{x}_i)\right) + \left(e^{-\beta}\right) \sum_{i=1}^{N} w_{m,i}\right)$
- 得到$\phi_m(\mathbf{x}) = \underset{\phi}{\operatorname{argmin}} \sum_{i=1}^{N} w_{m,i}\, \mathbb{I}\left(y_i \neq \phi(\mathbf{x}_i)\right)$，即最佳的$\phi_m$为错误率最小的弱分类器。

推导过程同之前的$\frac{\partial Z_m}{\partial \alpha_m}$推导

- 将ϕ_m代入L_m，并令$\frac{\partial \mathrm{L}_m}{\partial \beta_m} = 0 \;\; ==> \beta_m = \frac{1}{2} log \frac{1-\varepsilon_m}{\varepsilon_m}$，
- 其中错误率$\varepsilon_m = \sum_{i=1}^{N} w_{m,i}\, \mathbb{I}\left(y_i \neq \phi_m(\mathbf{x}_i)\right) / \sum_{i=1}^{N} w_{m,i}$。

▶ 前向逐步递增—其他损失函数

- 指数损失对outliers比较敏感，而且也不是任何二值变量y的概率密度取log后的表示。
- 因此另一种选择是损失函数取负log似然损失，得到logitBoost.
- 对回归问题，损失函数可取L2损失，得到L2boosting

L2Boosting

- 对L2损失：$L(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$
  - 初始化： $f_0(\mathbf{x}) = \overline{y}$

- 在第m步，损失函数的形式为
- $L(f_{m-1}(\mathbf{x}_i) + \beta_m \phi_m(\mathbf{x}_i), y_i) = (f_{m-1}(\mathbf{x}_i) + \beta_m \phi_m(\mathbf{x}_i) - y_i)^2$
- $= \left(-(y_i - f_{m-1}(\mathbf{x}_i)) + \beta_m \phi_m(\mathbf{x}_i)\right)^2 = \left(r_{m,i} - \beta_m \phi_m(\mathbf{x}_i)\right)^2$
- 其中$r_{m,i} = f_{m-1}(\mathbf{x}_i) - y_i$
- 不失一般性，假设β=1，因此用弱学习器来预测残差$r_{m,i}$，称为L2Boosting。

Shrinkage

- 通常对系数增加一个小的收缩因子(XGBoost中称为学习率)，测试性能更好，即

$$f_m(\mathbf{x}_i) = f_{m-1}(\mathbf{x}_i) + \eta\beta_m\phi_m(\mathbf{x}_i)$$

- 其中$0 < \eta < 1$，通常取一个较小的值，如$\eta = 0.1$。
- 较小的收缩因子通常意味着更多弱分学习器。

Boosting as 函数梯度下降

- 前向逐步递增建模部分我们讨论了不同损失函数对应的boosting算法，其实可推导更一般的模型：gradient boosting

- 目标：$\min_{\mathbf{f}} \frac{1}{N}\sum_{i=1}^{\mathrm{N}} L(f(\mathbf{x}_i), y_i)$，
- 其中$\mathbf{f} = \{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)\}$ 为“参数”

- 优化：逐步梯度下降（stagewise，gradient boosting）

Gradient Boosting

- 目标：$\min_{\mathbf{f}} \frac{1}{N} \sum_{i=1}^{\mathrm{N}} L(f(\mathbf{x}_i), y_i)$，
- 在第m步，$\mathbf{f} = \mathbf{f}_{\mathrm{m}-1}$
- 梯度为：$g_{m,i} = \left[\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\right]_{\mathbf{f}=\mathbf{f}_{\mathrm{m}-1}}$
- 然后更新：$\mathbf{f} = \mathbf{f}_{\mathrm{m}-1} - \beta_m \mathbf{g}_m$, 其中$\mathbf{g}_m = \left(g_{m,1}, \dots, g_{m,N}\right)^T$
- 其中$\beta_m = \operatorname*{argmin}_{\beta} \sum_{i=1}^{\mathrm{N}} L\left(f_{\mathrm{m}-1}(\mathbf{x}_i) - \beta g_{m,i}, y_i\right)$为步长

Gradient Boosting

- 上述算法只在N个数据点优化**f**
- 将上述算法修改为用一个弱学习器近似负梯度，即
- $\phi_m = \underset{\phi}{\operatorname{argmin}} \sum_{i=1}^{N} \left(-g_{m,i} - \phi(\mathbf{x}_i)\right)^2$
- 对上述算法，损失函数取L2，得到L2Boosting
- 该一般框架对很多损失函数都适用
  - Logistic损失
  - Huber损失
  - …

▶ Gradient Boosting Algorithm

- 1. Initialize $f_0(\mathbf{x}) = \underset{f}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{\mathrm{N}} L\left(y_i, f(\mathbf{x}_i)\right)$
- 2. **for** m =1:M **do**
- 3. Compute the gradient residual using $r_{m,i} = -\left[\frac{\partial L\left(y_i, f(\mathbf{x}_i)\right)}{\partial f(\mathbf{x}_i)}\right]_{\mathbf{f}=\mathbf{f}_{\mathbf{m}-1}}$
- 4. Use the weak learner which minimizes $\sum_{i=1}^{\mathrm{N}} \left(r_{m,i} - \phi_m(\mathbf{x}_i)\right)^2$
- 5. Update $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \eta\phi_m(\mathbf{x})$
- 6. return $f(\mathbf{x}) = f_M(\mathbf{x})$

▶ 大纲

- Boosting
- Gradient Boosting
- GBM in Scikit learn
- XGBoost
- LightGBM

Scikit-learn中的GBM

不止于代码

- 分类器：GradientBoostingClassifier

- sklearn.ensemble.**GradientBoostingClassifier**(*loss='deviance', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_split=1e-07, init=None, random_state=None, max_features=None, verbose=0, max_leaf_nodes=None, warm_start=False, presort='auto'*)
  - 由于弱学习器为CART，所以很多参数与树模型的参数相同
  - 额外的参数（蓝色）主要关于弱学习器组合

参数	说明
loss	待优化的目标函数，‘deviance’表示采用logistic损失，输出概率值；‘exponential’表示采用指数损失。缺省 ‘deviance’
learning_rate	学习率或收缩因子。学习率和迭代次数 / 弱分类器数目n_estimators相关。 缺省：0.1
n_estimators	当数 / 弱分类器数目. 缺省:100
subsample	学习单个弱学习器的样本比例。缺省为：1.0

CSDN
不 止 于 代 码

THANK YOU

AI100