Boosting

Ideas

Boosting 算法被认为是近年机器学习领域最有效的算法之一。

其核心思想为：

- 找到一系列弱分类器总是比找到一个强分类器容易的多。
- 将一系列弱分类器组合起来便有可能得到一个强分类器。

主要内容

- Adaboost算法讲解

- GBDT/GBRT算法讲解

一.Adaboost算法

- 数据: $\{(x_n, y_n)\}_{n=1}^{N}$, $x \in R^d$, $y \in \{-1,1\}$

- 目标: $G(x) = y$

一.Adaboost算法

1.对每个$x_n \in \boldsymbol{x}$, 初始化**权重**

$$D_1 = \{w_1, w_2, \ldots w_{\mathrm{n}}\}，\ w_{\mathrm{n}} = 1/N$$

2.对迭代次数m=1,2,3….M：

对具有权重D_1的数据集进行训练得到 弱分类器G_m

$$G_m(x) \rightarrow \{-1,+1\}$$

根据损失函数计算错误率ε_m:

$$\varepsilon_{\mathrm{m}} = [\sum \mathrm{D}_1(\mathrm{x}_{\mathrm{i}}) \cdot \mathrm{I}(\mathrm{y}_{\mathrm{i}} \neq G_m(\mathrm{x}_{\mathrm{i}}))] \,/\, [\sum G_m(\mathrm{x}_{\mathrm{i}})]$$

计算分类器权重α_m

$$\alpha_m = \log((1 - \varepsilon_m)/\varepsilon_m)$$

更新权重D:

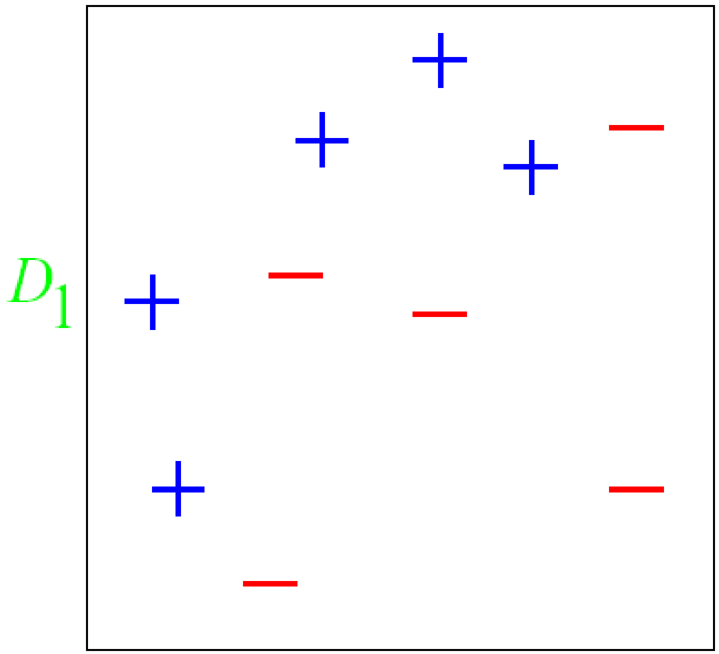
$$D_{m+1}(x_i) = D_m(x_i) \cdot \exp[\alpha_m \mathrm{I}(y_i \neq G_m(x_i))]$$

得到最终分类器：

$$G(x) = \mathrm{sign}\,[\sum \alpha_i G_i(x)]$$

一.Adaboost算法

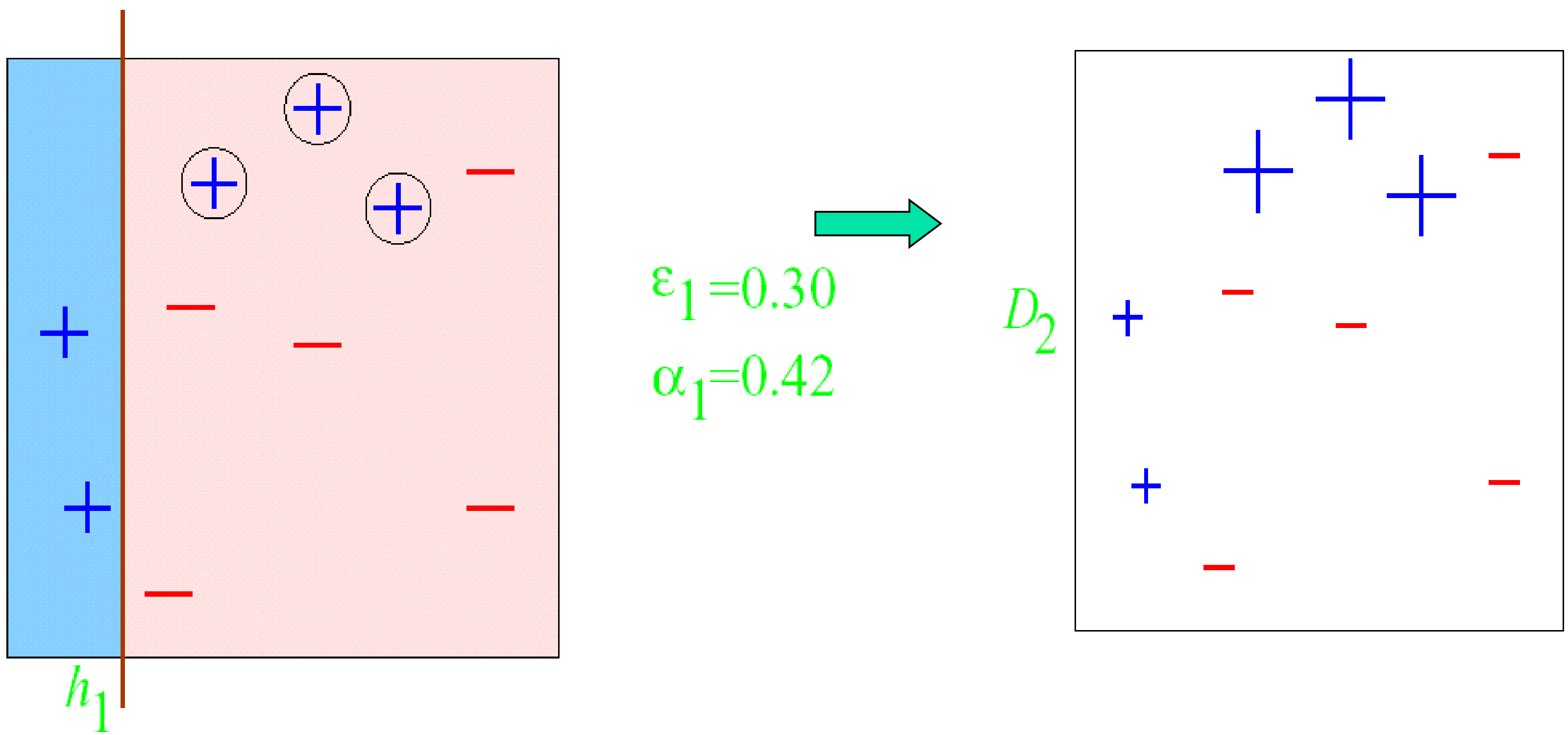
一个简单的例子



原始数据集:对每个数据赋予等值的权重

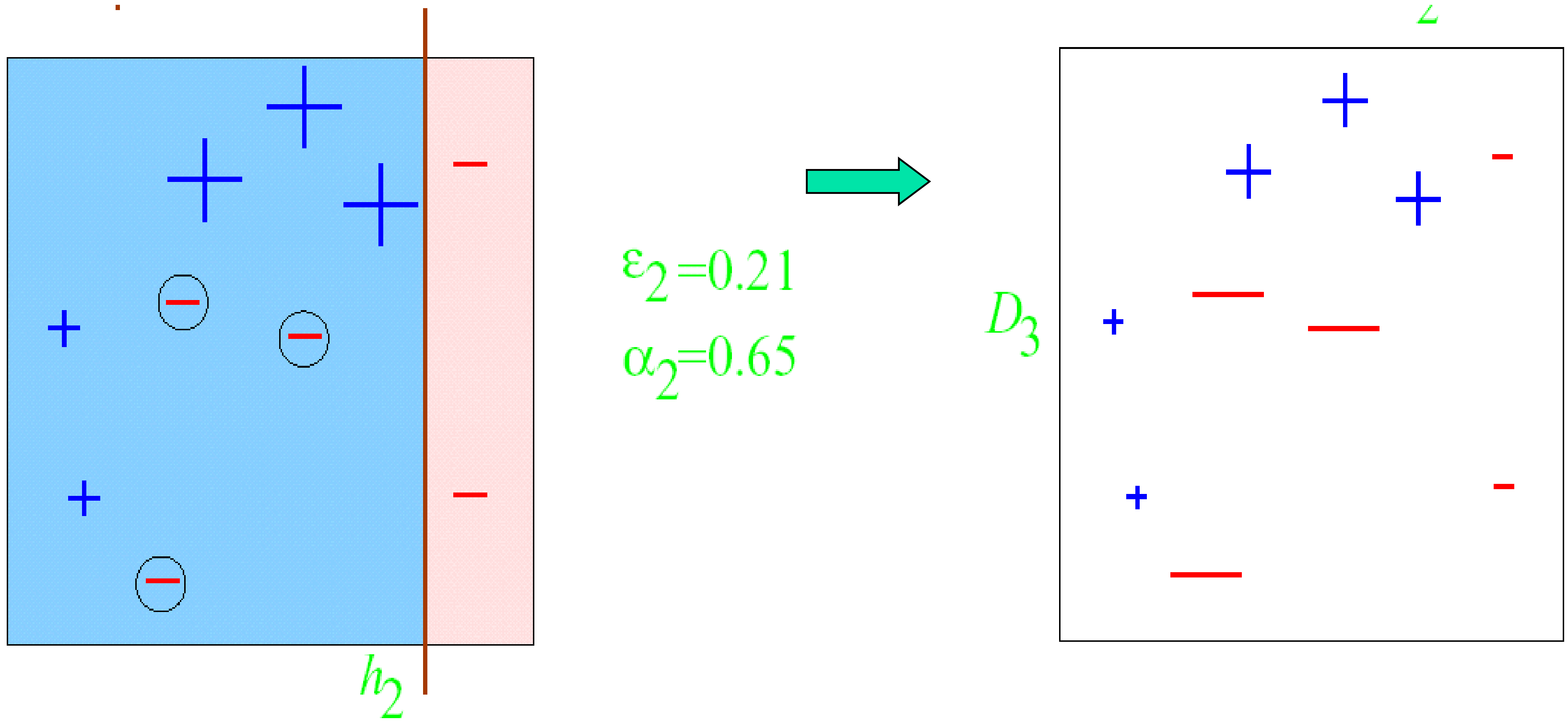
一.Adaboost算法

第一轮划分:



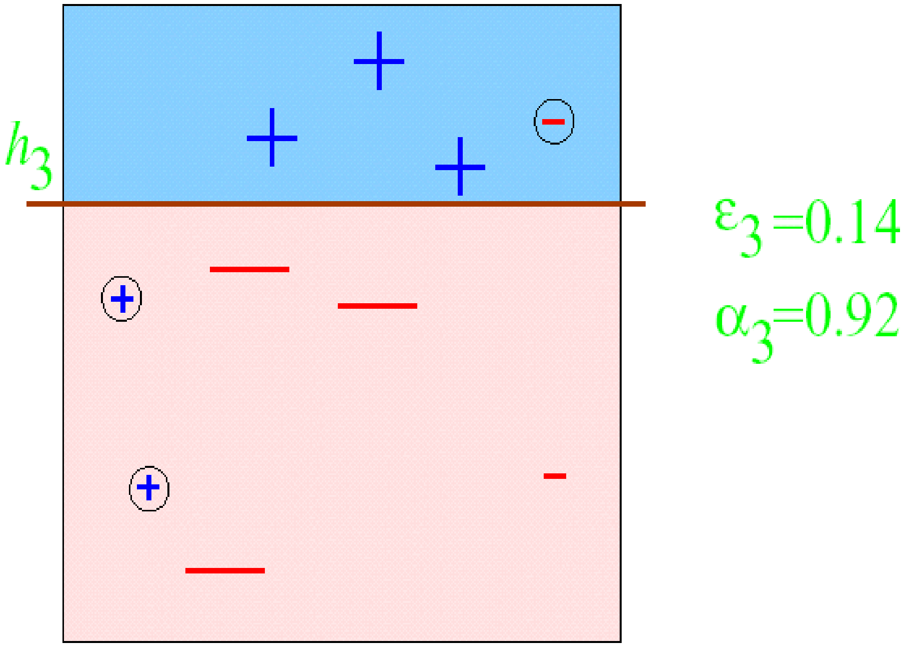
一.Adaboost算法

第二轮划分：



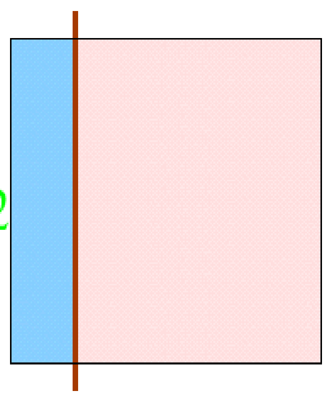
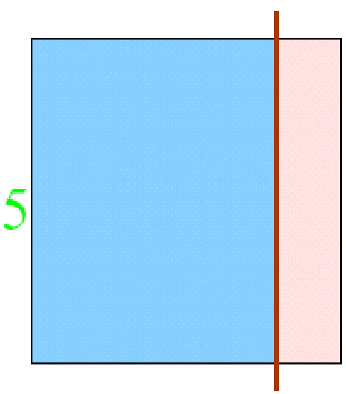
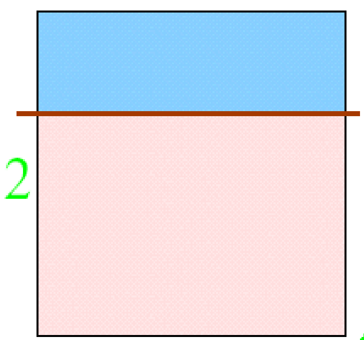
一.Adaboost算法

第三轮划分:



一.Adaboost算法

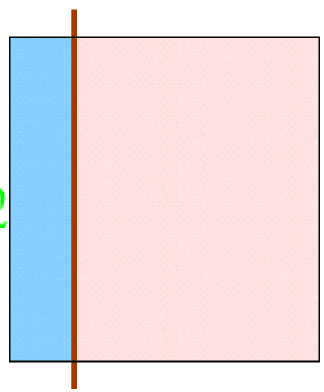
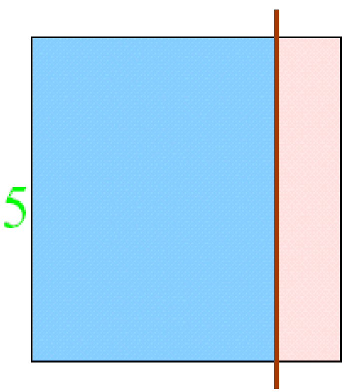
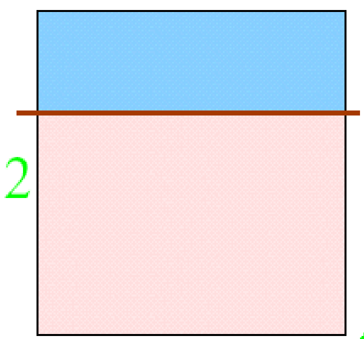
将弱分类器进行组合：

$H_{\text{final}} = \text{sign}\Big(0.42$  $+ 0.65$  $+ 0.92$  $\Big)$

一.Adaboost算法

将弱分类器进行组合：

H_{final}

$= \text{sign}\big($ 0.42  $+\ 0.65$  $+\ 0.92$  $\big)$

一.Adaboost算法

实际应用：

数据源：UCI数据库上的Adult数据集，数据量32000，划分27000作为训练集，5000作为测试集

数据一览：

```
25, Private, 226802, 11th, 7, Never-married, Machine-op-inspct, Own-child, Black, Male, 0, 0, 40, United-States, <=50K.
38, Private, 89814, HS-grad, 9, Married-civ-spouse, Farming-fishing, Husband, White, Male, 0, 0, 50, United-States, <=50K.
28, Local-gov, 336951, Assoc-acdm, 12, Married-civ-spouse, Protective-serv, Husband, White, Male, 0, 0, 40, United-States, >50K.
44, Private, 160323, Some-college, 10, Married-civ-spouse, Machine-op-inspct, Husband, Black, Male, 7688, 0, 40, United-States, >50K.
18, ?, 103497, Some-college, 10, Never-married, ?, Own-child, White, Female, 0, 0, 30, United-States, <=50K.
34, Private, 198693, 10th, 6, Never-married, Other-service, Not-in-family, White, Male, 0, 0, 30, United-States, <=50K.
29, ?, 227026, HS-grad, 9, Never-married, ?, Unmarried, Black, Male, 0, 0, 40, United-States, <=50K.
63, Self-emp-not-inc, 104626, Prof-school, 15, Married-civ-spouse, Prof-specialty, Husband, White, Male, 3103, 0, 32, United-States, >50I
24, Private, 369667, Some-college, 10, Never-married, Other-service, Unmarried, White, Female, 0, 0, 40, United-States, <=50K.
55, Private, 104996, 7th-8th, 4, Married-civ-spouse, Craft-repair, Husband, White, Male, 0, 0, 10, United-States, <=50K.
65, Private, 184454, HS-grad, 9, Married-civ-spouse, Machine-op-inspct, Husband, White, Male, 6418, 0, 40, United-States, >50K.
36, Federal-gov, 212465, Bachelors, 13, Married-civ-spouse, Adm-clerical, Husband, White, Male, 0, 0, 40, United-States, <=50K.
26, Private, 82091, HS-grad, 9, Never-married, Adm-clerical, Not-in-family, White, Female, 0, 0, 39, United-States, <=50K.
58, ?, 299831, HS-grad, 9, Married-civ-spouse, ?, Husband, White, Male, 0, 0, 35, United-States, <=50K.
48, Private, 279724, HS-grad, 9, Married-civ-spouse, Machine-op-inspct, Husband, White, Male, 3103, 0, 48, United-States, >50K.
```

一.Adaboost算法

选择弱分类器：

基于单特征的决策树桩：网格式搜索判断最优阈值

```
def losscalculate(self,predict, weight, label):  # em计算函数
    loss = 0
    if len(predict) != len(weight):
        raise IndexError
    for pred, wgt, lb in zip(predict, weight, label):
        loss += wgt * self.I(pred, lb)
    return loss  # 计算弱分类器loss
```

```
for j in np.arange(rangeMin, rangeMax, step):
    loss = self.losscalculate(self.DecisionStumppredict(dataMatrix, j), weightmatrix, classLabels)
    if loss < minError:
        bestError = loss
        bestkey = j
```

一.Adaboost算法

第一步:初始化权重

```
weight = np.zeros(data.shape[0])
weight.fill(1 / data.shape[0])
```

第二步:进入循环，对每个特征构建弱分类器

```
stump = self.buildStump(data[:, w], weight, label,stepsize)
```

第三步:计算分类器的权重

```
Alpha = self.alpha(stump[0])

def alpha(self,loss):#计算alpha
    return 1 / 2 * math.log((1 - loss) / loss)
```

一.Adaboost算法

第四步:更新权重参数

```
predict = self.DecisionStumppredict(data[:, w], stump[1])#弱分类器预测函数
Z = self.calZ(Alpha, label, predict, weight)
cnt = 0
for i, j, k in zip(weight, label, predict):
    weight[cnt] = i * math.exp(-Alpha * j * k)/Z#权重更新
    cnt += 1
```

第五步:得到强分类器

```
def predict(self,data, stumps, alphas):
    Pred = np.zeros(data.shape[0])

    for i in range(data.shape[1]):
        cnt = 0
        pred = self.DecisionStumppredict(data[:, i], stumps[i][1])
        for j, k in zip(pred, Pred):
            Pred[cnt] += j * alphas[i]#加权累计各个弱分类器的预测结果
            cnt += 1
```

二.GBDT/GBRT算法

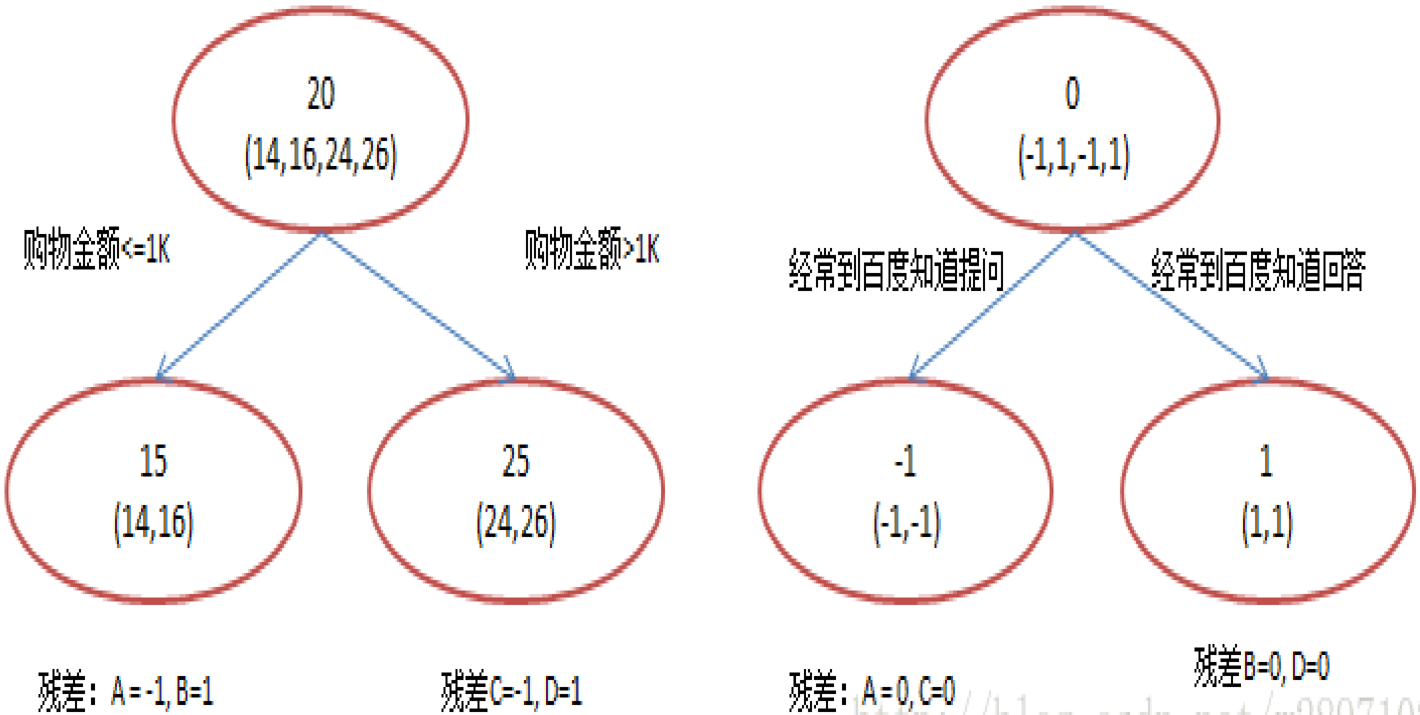
GBDT(Gradient Boosting Decision Tree) ：是一种迭代的决策树算法，

该算法由多棵决策树组成，每一颗决策树学习的是之前决策树的**残差**，

所有树的结论累加起来做最终答案。

二. GBDT/GBRT算法

什么是残差？



二. GBDT/GBRT算法

传统的BOOST与Gradient Boosting的区别：

Gradient Boost：每个新的模型的建立是为了使得之前模型的残差往梯度方向减少。

传统Boost：对正确、错误的样本进行加权（每一步结束后，增加分错的点的权重，减少分对的点的权重）。

二. GBDT/GBRT算法

核心问题：优化问题

$$argmin \sum_{i=1}^{N} L(y_i, c)$$

核心方法：梯度下降残差近似方法

$$r_{mi} = -\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}$$

GBRT算法流程

输入： $\{(\mathrm{x}_n, \mathrm{y}_n)\}_{n=1}^{N}, \mathrm{x} \in \mathrm{R}^d, y \in \{-1,1\}$

输出：回归树$\hat{f}(x)$

GBRT算法流程

1. 初始化

$$f_0(x) = argmin \sum_{i=1}^{N} L(y_i, c)$$

2. 对第1, 2, 3. . . M次迭代

a. 对第1, 2, 3. . . N计算

$$r_{mi} = -\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}$$

b. 对r_{mi}拟合一颗回归树，得到第m棵树的叶节点区域$R_{m\mathrm{j}}$

c. 对j=1, 2, . . . J 计算

$$c_{mj} = argmin \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + c)$$

d. 更新$f_{\mathrm{m}}(x)$=$f_{\mathrm{m}-1}(x)$+$\sum_{\mathrm{j}=1}^{J} c_{mj}\, I(x \in R_{mJ})$

3. 得到回归树$\hat{f}(x) = f_M(x)$

GBDT与GBRT的区别

GBDT的损失函数主要为deviance loss：f(x)为叶节点的对数输出

$$
\begin{aligned}
L(y, p(x)) &= -\sum_{k=1}^{K} I(y = \mathcal{G}_k) \log p_k(x) \\
&= -\sum_{k=1}^{K} I(y = \mathcal{G}_k) f_k(x) + \log\left(\sum_{\ell=1}^{K} e^{f_\ell(x)}\right). \quad (10.22)
\end{aligned}
$$

$$
p_k(x) = \frac{e^{f_k(x)}}{\sum_{l=1}^{K} e^{f_l(x)}}, \qquad (10.21)
$$

GBRT的损失函数主要为MSE（平方均值函数）

谢谢观看！