梯度提升树

1 提升树

1.1 前向分步算法

考虑加法模型：

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m)$$

其中 $b(x; \gamma_m)$ 为基函数，γ_m 为基函数的参数，β_m 为基函数的系数。

在给定训练数据和损失函数 $L(y, f(x))$ 时，学习加法模型 $f(x)$ 成为最小化以下经验风险极小化问题：

$$\min_{\beta_m, \gamma_m} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} \beta_m b(x_i; \gamma_m))$$

这个问题有 $2M$ 个参数，直接优化较为复杂。前向分步算法求解这一优化问题的思想是：从前向后不断学习，每一步只学习一个基函数及其系数，简化其复杂度。即每步只需优化以下函数：

$$\min_{\beta, \gamma} \sum_{i=1}^{N} L(y_i, \beta b(x_i; \gamma))$$

更一般地，参数 β_m, γ_m 可以如下求得：

$$(\beta_m, \gamma_m) = arg \min_{\beta, \gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$$

并且由求出的 β_m, γ_m 更新 $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$。

这样，前向分步算法将同时求解 $m = 1, 2, ..., M$ 的所有参数 β_m, γ_m 的优化问题简化为**逐次求解**各个 β_m, γ_m 的优化问题。事实上，AdaBoost 就是一种前向分步算法。

1.2 提升树

以决策树为基函数的提升方法称为提升树。提升树可以表示为决策树的加法模型：

$$f_M(x) = \sum_{m=1}^{M} T(x, \Theta_m)$$

其中，$T(x, \Theta_m)$ 表示决策树，Θ_m 为决策树的参数，M 为树的个数。使用前向分步算法，以 $f_0(x) = 0$ 为初始提升树，第 m 步的模型是：

$$f_m(x) = f_{m-1}(x) + T(x, \Theta_m)$$

当前已求得的模型为 $f_{m-1}(x)$，Θ_m 可由经验风险最小化确定：

$$\hat{\Theta}_m = arg \min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m))$$

1.3 梯度提升树

以回归提升树为例，其使用平方误差损失函数 $L(y, f(x)) = (y - f(x))^2$。其损失变为：

$$\begin{aligned} L(y, f_{m-1}(x) + T(x; \Theta_m)) &= [y - f_{m-1}(x) - T(x; \Theta_m)]^2 \\ &:= [r - T(x; \Theta_m)]^2 \end{aligned}$$

这里 $r = y - f_{m-1}(x)$ 是当前模型的拟合残差。因此回归提升树只需简单拟合当前模型的残差。回归提升树算法如下：

输入：训练数据集 $T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}, x_i \in R^n, y_i \in R$ 和树的棵树 M。

1. 初始化 $f_0(x) = 0$
2. 对 $m = 1, 2, ..., M$

- 计算残差 $r_{mi} = y_i - f_{m-1}(x_i), \quad i = 1, 2, ..., N$
- 对残差 r_{mi} 学习一个回归树 $T(x; \Theta_m)$
- 更新 $f_m(x) = f_{m-1}(x) + T(x; \Theta_m)$

3. 得到回归问题的提升树 $f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m)$

输出：提升树 $f_M(x)$。

回归提升树是一种特殊的梯度提升树。梯度提升树的主要思想是**不断地对数据的残差进行拟合**。但是很多时候，一般的损失函数的优化较为困难。我们可以使用损失函数的负梯度在当前模型的值来近似计算残差：

$$r_{mi} = -[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}]_{f(x)=f_{m-1(x)}}$$

并对 r_{mi} 继续拟合一个决策树。回归提升树是梯度提升树的一种特例，因为：

$$-\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}|_{f_{m-1(x)}} = -\frac{\partial [(y_i - f(x_i))^2]}{\partial f(x_i)}|_{f_{m-1(x)}} = 2(y_i - f_{m-1}(x_i))$$

因此，$r = y - f_{m-1}(x)$。

2 代码实现

考虑以下数据：

x	y
1	5.56
2	5.70
3	5.91
4	6.40
5	6.80
6	7.05
7	8.90
8	8.70
9	9.00
10	9.05

2.1 sklearn 实现

准备数据：

```
import numpy as np
X = np.array([i+1 for i in range(10)]).reshape(-1,1)
Y = np.array([5.56,5.70,5.91,6.40,6.80,7.05,8.90,8.70,9.00,9.05]).reshape(-1,1)
```

梯度提升回归树的接口位于 sklearn.ensemble.GradientBoostingRegressor, 其文档可见**此处**。其中较为重要的参数有：

- n_estimators：基函数个数
- loss,criterion：优化函数和决策树切割标准
- 部分决策树的参数

```
from sklearn.ensemble import GradientBoostingRegressor
clf = [GradientBoostingRegressor(n_estimators=i+1,max_depth=1) for i in range(8)]
for c in clf:
    c.fit(X,Y.reshape(-1))
```

```
for c in clf:
    print('基分类器个数: ',clf.index(c)+1,'MSE: ',round(np.sum((c.predict(X)-Y.reshape(-1))**2),3))
```

```
## 基分类器个数:  1 MSE:  15.849
## 基分类器个数:  2 MSE:  13.205
## 基分类器个数:  3 MSE:  11.062
## 基分类器个数:  4 MSE:  9.327
## 基分类器个数:  5 MSE:  7.906
## 基分类器个数:  6 MSE:  6.722
## 基分类器个数:  7 MSE:  5.706
## 基分类器个数:  8 MSE:  4.874
```

2.2 回归提升树

使用回归提升树时，需要使用到 CART 回归树，需要将其导入。

```
import numpy as np
from CART import RegressionTree

class BoostingTree:

    def __init__(self,n_estimators=10,max_depth=1):
        self.n_estimators = n_estimators
        self.max_depth = max_depth

    def fit(self,X,Y):
        self.clf_list = []
        for m in range(self.n_estimators):
            clf = RegressionTree(max_depth = self.max_depth)
            clf.fit(X,Y)
            self.clf_list.append(clf)
```

```
            Y = Y-clf.predict(X)

    def predict(self,new_X):
        return np.sum(np.array([clf.predict(new_X) for clf in self.clf_list]),axis=0)
```

结果如下：

```
clf = [BoostingTree(n_estimators=i+1,max_depth=1) for i in range(6)]
for c in clf:
    c.fit(X,Y)

for c in clf:
    print('基分类器个数：',clf.index(c)+1,'MSE：',round(np.sum((c.predict(X)-Y)**2),3))
```

```
## 基分类器个数： 1 MSE： 1.93
## 基分类器个数： 2 MSE： 0.801
## 基分类器个数： 3 MSE： 0.478
## 基分类器个数： 4 MSE： 0.306
## 基分类器个数： 5 MSE： 0.229
## 基分类器个数： 6 MSE： 0.172
```