GBDT

加法模型

在Adaboost模型中，我们把每个基本分类器合成一个复杂分类器的方法是每个基本分类器的加权和，也即$f(x)=\sum\limits_{m=1}^{M}\beta_m b\left(x;\gamma_m\right)$。这类模型也被称为加法模型，其中，$b\left(x;\gamma_m\right)$称为基函数，$\gamma_m$称为基函数的参数，$\beta_m$称为基函数的系数，$\beta_m$表示对应的基函数在加法模型$f(x)$中的重要性。

在给定训练数据及损失函数$L(y,f(x))$的条件下，学习加法模型$f(x)$成为经验风险极小化问题，也即：

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

这是一个复杂的优化问题，很难通过简单的凸优化的相关知识进行解决。我们可以使用向前分布算法来解决这种形式的问题。其基本思路是：从前向后每一步只优化一个基函数及其系数，逐步逼近目标函数，从而降低优化的复杂度。也即：

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

向前分布算法

给定数据集$T=\left\{\left(x_1,y_1\right),\left(x_2,y_2\right),\cdots,\left(x_N,y_N\right)\right\}$，$x_i\in\mathcal{X}\subseteq\mathbf{R}^n$，$y_i\in\mathcal{Y}=\{+1,-1\}$。损失函数$L(y,f(x))$，基函数集合$\{b(x;\gamma)\}$，我们需要输出加法模型$f(x)$。

- 初始化：$f_0(x)=0$
- 对$m=1,2,\ldots,M$:
  - (a) 极小化损失函数：

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

    得到参数β_m与γ_m
  - (b) 更新：

    $$f_m(x)=f_{m-1}(x)+\beta_m b\left(x;\gamma_m\right)$$
- 得到加法模型：

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

这样，前向分步算法将同时求解从m=1到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的问题。

梯度提升决策树(GBDT)

基于残差学习的提升树算法

- 基函数：回归（决策）树
- 基分类器的系数：每个样本的残差

输入数据集$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\},x_i\in\mathcal{X}\subseteq\mathbf{R}^n,y_i\in\mathcal{Y}\subseteq\mathbf{R}$，输出最终的提升树$f_M(x)$

- 初始化$f_0(x)=0$
- 对$m=1,2,\ldots,M$:
  - 计算每个样本的残差:$r_{mi}=y_i-f_{m-1}(x_i),\quad i=1,2,\cdots,N$
  - 拟合残差r_{mi}学习一棵回归树，得到$T(x;\Theta_m)$
  - 更新$f_m(x)=f_{m-1}(x)+T(x;\Theta_m)$
- 得到最终的回归问题的提升树：$f_M(x)=\sum_{m=1}^{M}T(x;\Theta_m)$

梯度提升决策树

拟合残差只能适用于损失函数为平方损失的特殊情况，而对于更复杂的损失函数（例如加入了正则化项），就不能简单地拟合残差了。为了将模型扩展到更复杂的损失函数中，Freidman提出了梯度提升算法(gradient boosting)，利用损失函数的负梯度在当前模型的值$-\left[\frac{\partial L(y,f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$作为回归问题提升树算法中的残差的近似值，拟合回归树。具体模型如下：

输入训练数据集$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\},x_i\in\mathcal{X}\subseteq\mathbf{R}^n,y_i\in\mathcal{Y}\subseteq\mathbf{R}$和损失函数$L(y,f(x))$，输出回归树$\hat{f}(x)$

- 初始化$f_0(x)=\arg\min_c\sum_{i=1}^{N}L(y_i,c)$
- 对于$m=1,2,\ldots,M$:
  - 对$i=1,2,\ldots,N$计算：$r_{mi}=-\left[\frac{\partial L(y_i,f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$
  - 对r_{mi}拟合一个回归树，得到第m棵树的叶结点区域$R_{mj},j=1,2,\cdots,J$
  - 对$j=1,2,\ldots J$，计算：$c_{mj}=\arg\min_c\sum_{x_i\in R_{mj}}L(y_i,f_{m-1}(x_i)+c)$
  - 更新$f_m(x)=f_{m-1}(x)+\sum_{j=1}^{J}c_{mj}I(x\in R_{mj})$
- 得到回归树：$\hat{f}(x)=f_M(x)=\sum_{m=1}^{M}\sum_{j=1}^{J}c_{mj}I(x\in R_{mj})$

GBDT的实践

基于boston房价数据进行实践。

导入相关包

```
import pandas as pd
from sklearn import datasets
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

读入数据

```
boston = datasets.load_boston()
X = boston.data
y = boston.target
features = boston.feature_names
boston_data = pd.DataFrame(X,columns=features)
boston_data["Price"] = y
```

构建模型

```
lr = LinearRegression()
dt = DecisionTreeRegressor(criterion="mse", min_samples_leaf=10)
gbdt = GradientBoostingRegressor(n_estimators=500, learning_rate=0.05,
random_state=0, loss='ls')
models = [("lr", lr), ("dt", dt), ("gbdt", gbdt)]
```

结果对比

```
def evaluate_model(model, X, y):
    score = cross_val_score(model, X, y, scoring='r2', cv=5,
error_score='raise')
    return score

for (name, model) in models:
    score = evaluate_model(model, X, y)
    print(f"Model:{name}; Mean: {score.mean():.3f}; Std: {score.std():.3f}")
```

> Model:lr; Mean: 0.353; Std: 0.377
> Model:dt; Mean: 0.411; Std: 0.347
> Model:gbdt; Mean: 0.665; Std: 0.159

可以看到，集成学习模型比简单的基学习器的效果有很大的提升。