IML Lab3 report

姓名：黄瑞轩　　学号：PB20111686

实验目的

以决策树为基训练一个 XGBoost 模型。

实验原理

回归树算法

本实验采用回归树为基本算法，定义回归树 f 惩罚值为

$$penalty(f) = \gamma T + \frac{1}{2}\lambda \|\boldsymbol{w}\|_2^2 \tag{1}$$

其中 γ, λ 为可调整的超参数，T 为叶子数，$\boldsymbol{w}$ 为回归树 f 的权重向量。

在学习第 t 个回归树时，定义待优化目标函数

$$\begin{aligned} Obj^{(t)} &= \sum_{i=1}^{n} loss(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} penalty(f_k) \\ &= \sum_{i=1}^{n} loss(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + penalty(f_t) + C \end{aligned}$$

其中 n 表示样本数量，$loss$ 是损失函数，对于我们的回归问题，取

$$loss(y_i, \hat{y}_i^{(t)}) = (y_i - \hat{y}_i^{(t)})^2 \tag{2}$$

将 $loss(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))$ 在 $\hat{y}_i^{(t-1)}$ 处泰勒展开可得

$$loss(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) \approx loss(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \tag{3}$$

其中 $g_i = \frac{\partial\, loss(y_i, \hat{y}_i^{(t-1)})}{\partial\, \hat{y}_i^{(t-1)}}, h_i = \frac{\partial^2 loss(y_i, \hat{y}_i^{(t-1)})}{\partial\, (\hat{y}_i^{(t-1)})^2}$，即 g_i 为一阶导数，h_i 为二阶导数。

其中，g_i 为一阶导数，h_i 为二阶导数，具体表达式如下：

$$g_i = \frac{\partial\, loss(y_i, \hat{y}_i^{(t-1)})}{\partial\, \hat{y}_i^{(t-1)}} = -2(y_i - \hat{y}_i^{(t-1)})$$
$$h_i = \frac{\partial^2\, loss(y_i, \hat{y}_i^{(t-1)})}{(\partial\, \hat{y}_i^{(t-1)})^2} = 2$$

由于学习第 t 个模型时，$loss(y_i, \hat{y_i}^{(t-1)})$ 是一个固定值，并且记分配到第 j 个叶子节点的样本集合为 I_j，因此只需要优化

$$\begin{aligned} Obj^{(t)} &= \sum_{i=1}^{n}\left[loss(y_i, \hat{y_i}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + penalty(f_t) + C \\ &= \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + penalty(f_t) \\ &= \sum_{j=1}^{T}\left[G_j w_j + \frac{1}{2}(H_j + \lambda) w_j^2\right] + \gamma T \end{aligned}$$

其中 $G_j = \sum_{i \in I_j} g_i, H_j = \sum_{i \in I_j} h_i$。求 $\frac{\partial\, Obj^{(t)}}{\partial\, w_j} = 0$ 可解得

$$w_j^\star = -\frac{G_j}{H_j + \lambda} \tag{4}$$

解出诸 $w_j^\star$ 后，目标值即

$$Obj^{(t,\star)} = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T \tag{5}$$

在训练的过程中，一开始所有样本都划分给根节点，若在满足后文所述的划分条件时，需要对所有属性划分的 Obj 增益做计算取最优，并且每个属性需要对各取值做试探取最优，写成算法就是

算法：选择最优划分属性和划分值
输入：该节点分配到的样本集合I_j
输出：选择的划分属性a，以及划分阈值$v(a)$

$F =$ 可用来划分的特征集合
for a **in** F :
    $V = I_j$中的节点在属性a上的升序取值集合
    for v **in** V :
        按照v划分I_j为$I_{jL} \cup I_{jR}$
        计算$Gain(I_{jL}, I_{jR})$
return $Gain(I_{jL}, I_{jR})$最大时的a, v

其中

$$
\begin{aligned}
Gain(I_{jL}, I_{jR}) &= Obj_1 - Obj_2 \\
Obj_1 &= -\frac{1}{2}\frac{G_j^2}{H_j + \lambda} + \gamma \\
Obj_2 &= -\frac{1}{2}\left[\frac{G_{jL}^2}{H_{jL} + \lambda} + \frac{G_{jR}^2}{H_{jR} + \lambda}\right] + 2\gamma
\end{aligned}
$$

为了决策一个节点是否需要继续划分，超参数 η 表示树深阈值，ζ 表示样本数阈值，当且仅当

$$
depth < \eta \textbf{ and } |I| > \zeta \tag{6}
$$

为 True 时才进行划分，否则递归终止，设置为叶子节点。

XGBoost

XGBoost 是加法模型，第 t 个基本模型的输出为

$$
\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{7}
$$

其中 $f_t(x)$为第 t 个基本模型，x_i 为第 i 个训练样本，y_i 表示第 i 个样本的真实标签，$\hat{y}_i^{(t-1)}$ 表示前 $t-1$ 个模型对第 i 个样本的标签最终预测值。

为了决策是否需要进行下一个基本模型的训练，超参数 μ 为基本模型数量上限，当且仅当

$$
t < \mu \tag{8}
$$

为 True 时才进行下一个基本模型的训练，否则迭代终止，最后一个基本模型即为输出的最终模型。

XGBoost 框架不包含读取数据方法，由框架的使用者实现，只需要向框架传入数据集即可。

实验步骤

使用 pandas 库读取 `train.data`，随机打乱后，抽取比例为超参数 $p(0 < p < 1)$ 的样本作为训练集，剩余样本作为测试集用以测试模型精度。

为了方便后续计算，训练集和测试集使用 numpy 数组。

回归树的叶子节点定义如下：

```
class Node:
    def __init__(self):
        self.splitFeature = None     # 用于划分的属性序号
        self.splitValue = None       # 用于划分的属性分界值

        self.left:Node = None        # 左孩子
        self.right:Node = None       # 右孩子

        self.leaf = False            # 指示是否叶子节点
        self.w = None                # 节点权重
```

回归树定义如下：

```
class RegressionTree:
    # 初始化回归树
    def __init__(self, y_hat=None):
        self.checker_depth = 3       # 最多迭代深度
        self.checker_num = 10        # 数据最多数目

        self.penalty_lambda = 0.1    # 惩罚系数 lambda
        self.penalty_gamma = 0.1     # 惩罚系数 gamma
        self.y_hat = y_hat

    # 获取某一划分的目标函数值
    def _get_split_score(self, y, y_hat):    # 按 Obj_1 计算

    # 计算权重
    def calcw(self, y, y_hat):               # 按公式（4）计算

    # 获得最佳 feature 和 split
    def _get_best_split(self, X, y, y_hat, depth):  # 按算法[选择最优划分属性和划分值]计算

    def _get_child_data(self, X, y, y_hat, a, v):   # 按划分结果划分数据

    def split(self, X, y, y_hat, depth):
        # ……计算划分结果
        if splitFeature is None:
            # 不需要划分，设置叶子，计算权重
        else:
            # ……划分后，递归计算左右子节点
```

```
            lChild = self.split("""……, """depth+1)
            rChild = self.split("""……, """depth+1)
            # ……
        return root

    # 训练一棵回归树
    def fit(self, X, y, y_hat):
        self.model = self.split(X, y, y_hat, 0)

    # 预测一个样本
    def _predict(self, x, model:Node):      # 采用迭代下降方法

    # 预测多条样本
    def predict(self, X):
        return np.array([self._predict(X[i], self.model) for i in
range(X.shape[0])])
```

XGBoost 框架定义如下：

```
class XGBoost:
    def __init__(self):
        self.iterUpperBound = 20    # 设置训练基本模型个数上限

    def fit(self, X, y):            # 训练模型，返回RMSE和R2变化列表

    def predict(self, X):           # 在预测集上运行，返回预测结果
        return self.model.predict(X)

    # 下面两个函数是从网上复制下来的，为的是更好的展现树结构
    def plot_tree_aux(self, g, node:Node, name):
    def plot_tree(self):
```

实现完毕后，采用如下步骤进行预测和可视化：

```
model = XGBoost()
RMSE, R2 = model.fit(X_train, y_train)
predictResults = model.predict(X_test)

# 输出测试集上 RMSE
model.plot_tree()    # 输出树结构
# 绘制 RMSE，R2 图
```

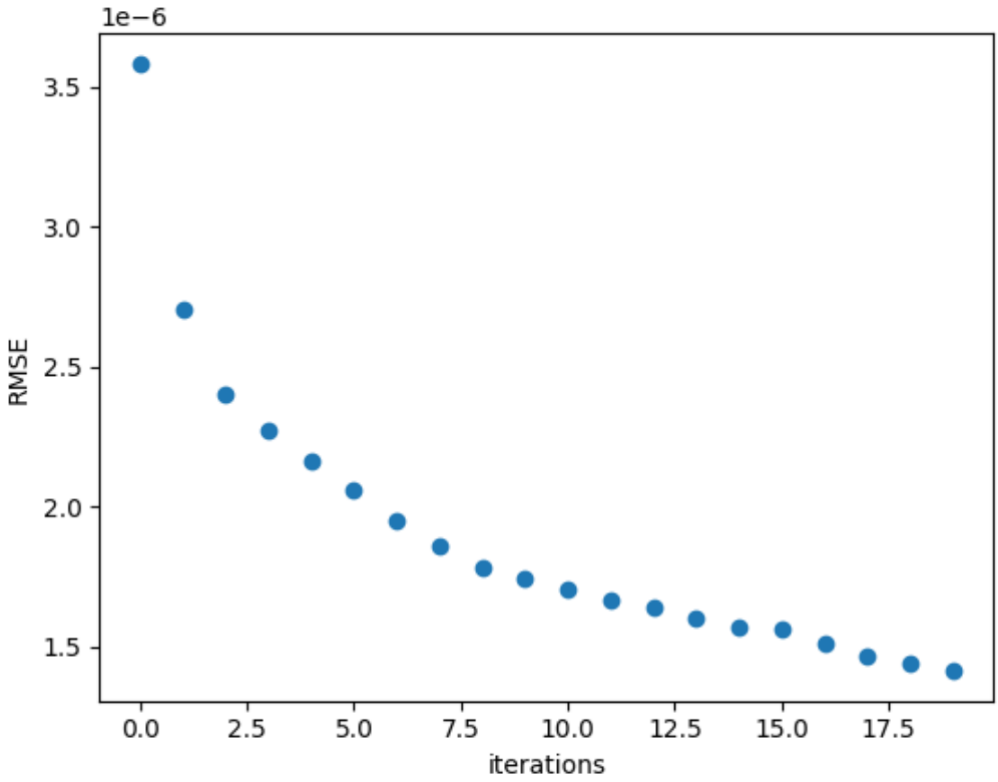
实验结果

本实验可供调整的超参数为：

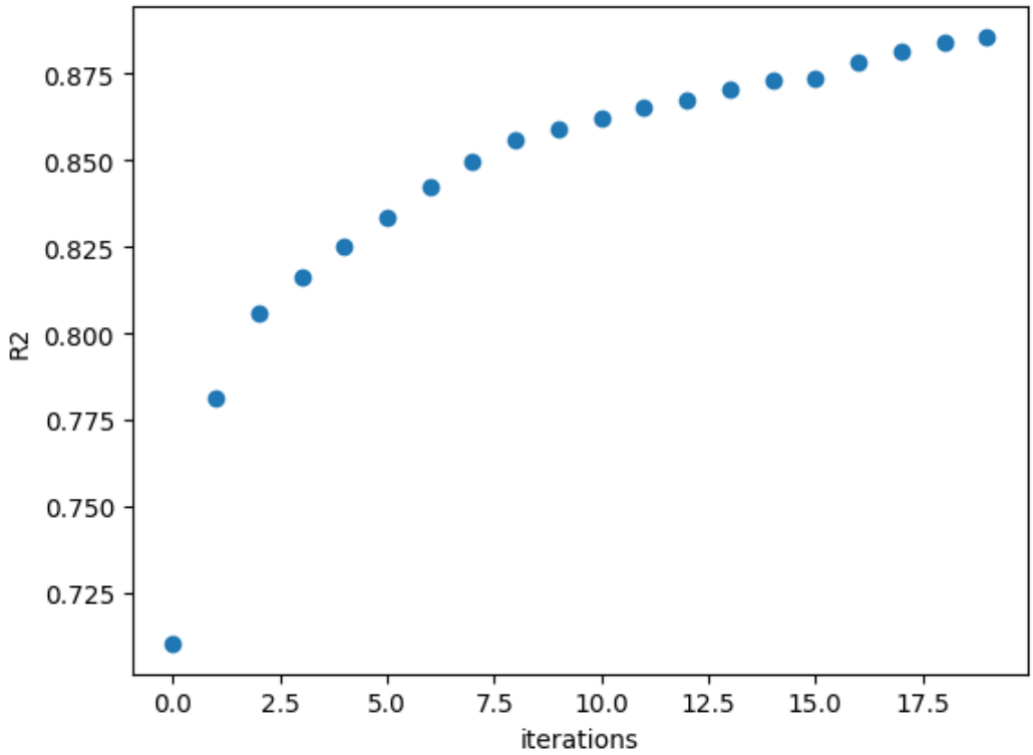
$$
\begin{aligned}
&p：\text{训练集选取比例}(p_0 = 0.7)\\
&\mu：\text{训练的基本模型数量上限}(\mu_0 = 20)\\
&\gamma：\text{惩罚项}(\gamma_0 = 0.1)\\
&\lambda：\text{惩罚项}(\lambda_0 = 0.1)\\
&\eta：\text{树的深度上限}(\eta_0 = 3)\\
&\zeta：\text{不再划分的样本数下限}(\zeta_0 = 10)
\end{aligned}
$$

对于其中的量 u，u_0 表示默认选项。

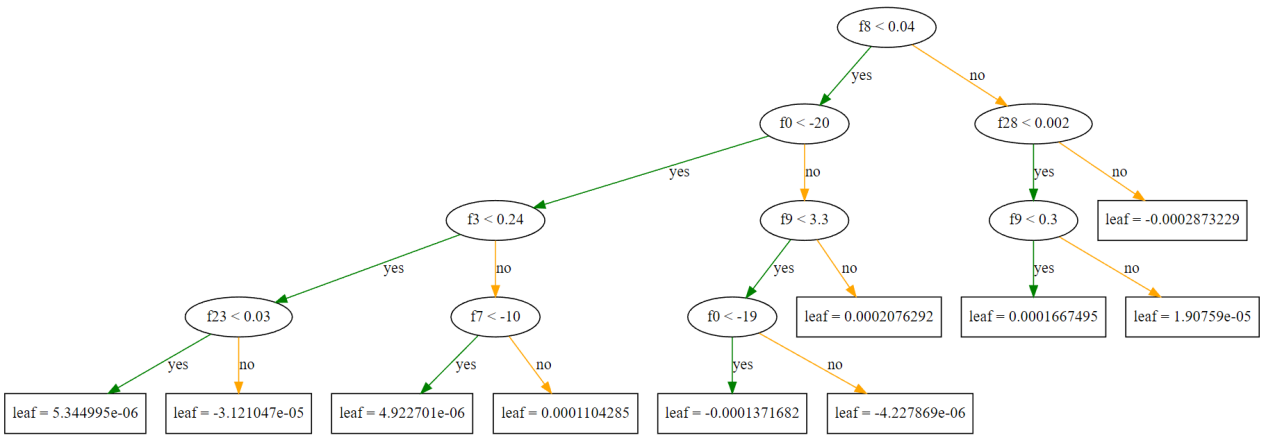
默认选项下实验结果



RMSE 随基本模型序号的改变



R2 随基本模型序号的改变



最后一个基本模型的结构

```
# 测试集上的输出
在测试集上运行的RMSE为： 4.144317333806676e-05
```

可见这个均方误差是比较好的，达到了训练的目的。

调整部分参数的对比

采用 RMSE 指标和训练时间指标，对比参数调整下的训练情况。

这里取两个比较关键的参数：训练的基本模型数量上限 μ，树的深度上限 η。在每个参数变化时，其他参数都与默认选项相同。

调整 μ 的训练指标对比

每个参数取值下的指标均是 3 次平行测试平均的结果。

μ	5	10	20	30	50
RMSE（10^{-5}）	4.285	4.262	4.328	4.352	4.523
训练时间（s）	9.68	17.84	41.16	60.15	105.93

当 μ 增大时，RMSE 呈现不太明显的增大趋势，这可能是因为训练次数过多导致了过拟合。

当 μ 增大时，训练时间呈现与 μ 的线性关系，说明每一次训练要做的计算量差不多是相同的。

调整 η 的训练指标对比

每个参数取值下的指标均是 3 次平行测试平均的结果。

η	2	3	4	5
RMSE（10^{-5}）	4.328	4.161	4.311	4.207
训练时间（s）	32.55	48.76	59.99	68.90

当 η 增大时，RMSE 的变化趋势不是很明显，可能是 XGBoost 性能较好，深度较小时也能工作地较好。

当 η 增大时，训练时间也增大，训练时间与 η 近似一种对数关系，可能是节点越深，需要划分的数据越少，相应的训练时间的增加就越少。