Machine Learning Lab3

Xiaoma

2022 年 11 月 20 日

实验要求

1. 实现决策树（回归树）的算法
2. 以决策树做基学习器实现 XGBoost 的算法

实验原理

XGBoost

XGBoost 是由多个基模型组成的一个加法模型，假设第 k 个基本模型是 $f_k(x)$，那么前 t 个模型组成的模型输出为

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

其中 x_i 为第 i 个训练样本，y_i 表示第 i 个样本的真实标签；$\hat{y_i^{(t)}}$ 表示前 t 个样本的标签最终预测值。

在学习第 t 个基模型时，XGBoost 要优化的目标函数为：

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

其中 n 表示训练样本的数量，$penalty(f_k)$ 表示第 k 个模型的复杂度的惩罚项，$loss(y_i, \hat{y}_i^{(t)})$ 表示损失函数。

对于回归问题：

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

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

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

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

则此时的优化目标变为

$$Obj^t = \sum_{i=1}^{n} [loss(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x) + \frac{1}{2} h_i f_t^2(x_i)] + penalty(f_t) + constant$$

去掉常数项 $loss(y_i, \hat{y}_i^{(t-1)})$ 和 $constant$，可得目标函数为

$$Obj^t = \sum_{i=1}^{n} [g_i f_t(x) + \frac{1}{2} h_i f_t^2(x_i)] + penalty(f_t)$$

决策树

假设决策树有 T 个叶子节点，每个叶子节点对应有一个权重。决策树模型就是将输入 x_i 映射到某个叶子节点，决策树模型的输出就是这个叶子节点的权重，即 $f_i(x_i) = w_{q(x_i)}$，w 是一个需要学的 T 维的向量，其中 $q(x_i)$ 表示把输入 x_i 映射的叶子节点的索引。

对于某一棵树，惩罚为

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

其中 $\gamma, lambda$ 为可调整的超参数，T 为叶子数，w 为权重向量。
将分配到第 j 个叶子节点的样本用 I_j 表示，即 $I_j = i|q(x_i) = j(1 \leq j \leq T)$。
综上，我们在树结构确定时，可以进行优化

$$\begin{aligned} Obj^t &= \sum_{i=1}^{n}[g_i f_t(x) + \frac{1}{2}h_i f_t^2(x_i)] + penalty(f_t) \\ &= \sum_{i=1}^{n}[g_i w_{q(x_i)} + \frac{1}{2}h_i w_{q(x_i)}^2] + \gamma T + \frac{1}{2}\lambda\|w\|^2 \\ &= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T \end{aligned}$$

简单期间，记 $G_i = \sum_{i\in I_j} g_i$，$H_i = \sum_{i\in I_j} h_i$

$$Obj_t = \sum_{j=1}^{T}[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2] + \gamma T$$

优化权重

在本次实验中

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

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

对 Obj^t 求偏导，令其为 0

$$\frac{\partial Obj^t}{\partial w_j} = G_j + (H_j + \lambda)w_j = 0$$

可以得到最优权重

$$w_j = -\frac{G_j}{H_j + \lambda}$$

$$\begin{aligned} Obj_t &= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)(-\frac{G_j}{H_j + \lambda}) + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)(-\frac{G_j}{H_j + \lambda})^2] + \gamma T \\ &= -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T \end{aligned}$$

实现过程

决策树

划分过程

对于每一棵决策树，划分节点的步骤为

1. 从根节点开始划分，初始情况下所有的训练样本 x_i 都分配给根节点
2. 随机选取某个特征，计算该特征下不同划分的收益
3. 选择收益最大的划分为本次划分
4. 若决策树达到最大深度或划分增益小于阈值，将该层节点设为叶子节点并设置权重

确定划分值的代码段为

```
    for value in values:
        splits = []
        sub_indexs1 = data.index[data[n] <= value].tolist()
        sub_indexs2 = data.index[data[n] > value].tolist()
        splits.append(sub_indexs1)
        splits.append(sub_indexs2)
        mark = 0
        for split in splits:
            if mark == 0:
                for idx in split:
                    Gleft += self.__g[0][idx]
                    Hleft += self.__h[idx]
                    mark += 1
            else:
                Gright = G - Gleft
                Hright = H - Hleft
        obj2 = -1 / 2 * (Gleft ** 2 / (Hleft + self.Lambda + 1e
            -16) + Gright ** 2 / (Hright + self.Lambda + 1e-16))
            + self.gamma
        Gain = obj1 - obj2
```

```
19            if Gain > best_Gain:
20                best_Gain = Gain
21                best_splits = splits
22                best_value = value
```

停止策略

1. 若树节点等于决策树最大深度，则停止划分
2. 划分后增益小于某个阈值则停止划分

XGBoost

训练过程

1. 随机选择特征训练一棵决策树并将其加入模型中
2. 根据验证集验证加入该决策树是否降低了 loss，否则将该决策树从模型中删去
3. 继续迭代直至停止

训练 XGBoost 的代码块为

```
1     for i in range(self.epoch):
2         tree = DecisionTree(max_depth = self.max_depth,
              min_splits=2, min_Gain=1e-10, gamma=self.gamma,
              Lambda=self.Lambda)
3         n = np.random.randint(0, 40)
4         #print("train epoch {}".format(i + 1))
5         #print("feature {}".format(n))
6         self.learning_rate = self.learning_rate * 0.95
7         tree.fit(x_train, y_train, temp_y, n, learning_rate=self
              .learning_rate)
8         self.trees.append(tree)
9         temp_y += tree.predict(x_train)
10        self.cal_RMSE(y_train, temp_y, flag='train')
```

```
11          if eval_test_x is not None:
12              eval_y += tree.predict(eval_test_x)
13              loss = self.cal_RMSE(eval_test_y, eval_y, flag='val'
                    )
14              R2 = self.R2(eval_test_y, eval_y)
15              self.R2s.append(R2)
16              if ESR:
17                  if loss < min_loss:
18                      min_loss = loss
19                      iter = 0
20                      #print("eval_test_loss {}".format(loss))
21                      #print("eval_test_R2 {}".format(R2))
22                  else:
23                      temp_y -= tree.predict(x_train)
24                      self.trees.pop()
25                      self.loss['val'].pop()
26                      self.loss['train'].pop()
27                      iter += 1
28                      if iter >= ESR:
29                          break
```

停止策略

1. 学习最多 epoch 棵决策树后停下来
2. 设置验证集，若选择 ESR 次训练得到的决策树都无法降低 loss，则停止

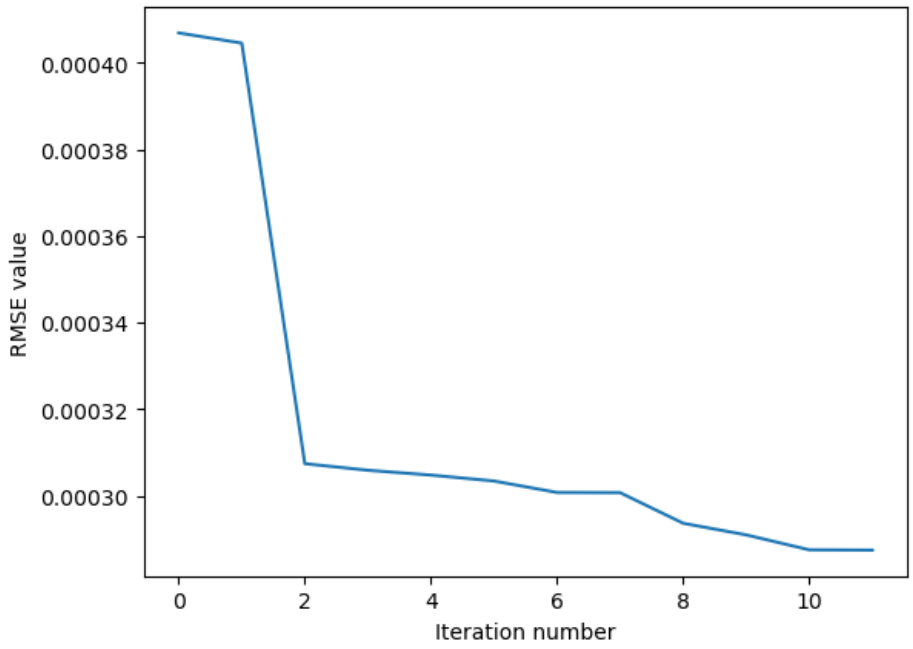
性能分析

优化前

在首次训练时并没有考虑到随机选择特征得到的决策树可能会使模型的 loss 增加，loss 会出现上升的情况。

此时训练过程中 loss 的变化为

loss 曲线



参数为

max_depth=30 learning_rate=1 gamma=0.1 Lambda=1.0

预测结果为

train RMSE=0.00027475 train R2=0.215212

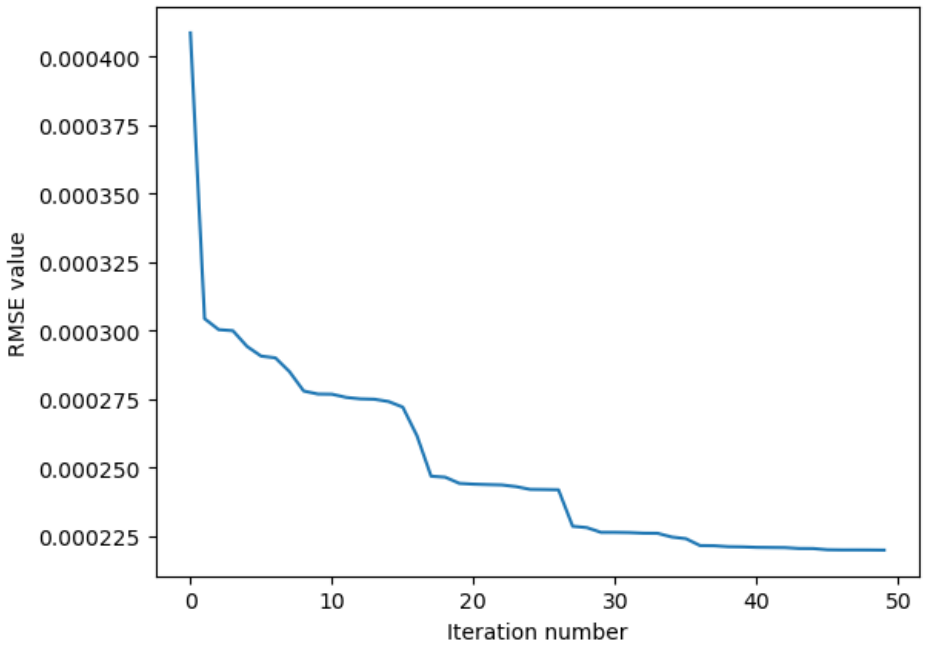
test RMSE=0.00034562 test R2=-0.18457

优化后

引入了新的 XGBoost 停止策略，并且判断根据验证集验证加入该决策树是否降低了 loss，否则将该决策树从模型中删去，确保了模型的 loss 可以一直下降，并且防止过早停止，使最终得到的模型性能更好。

此时训练过程中 loss 的变化为

loss 曲线



参数为

max_depth=30 learning_rate=1 gamma=0.1 Lambda=1.0

预测结果为

train RMSE=0.00021977 train R2=0.7076

test RMSE=0.0002770 test R2=0.540687

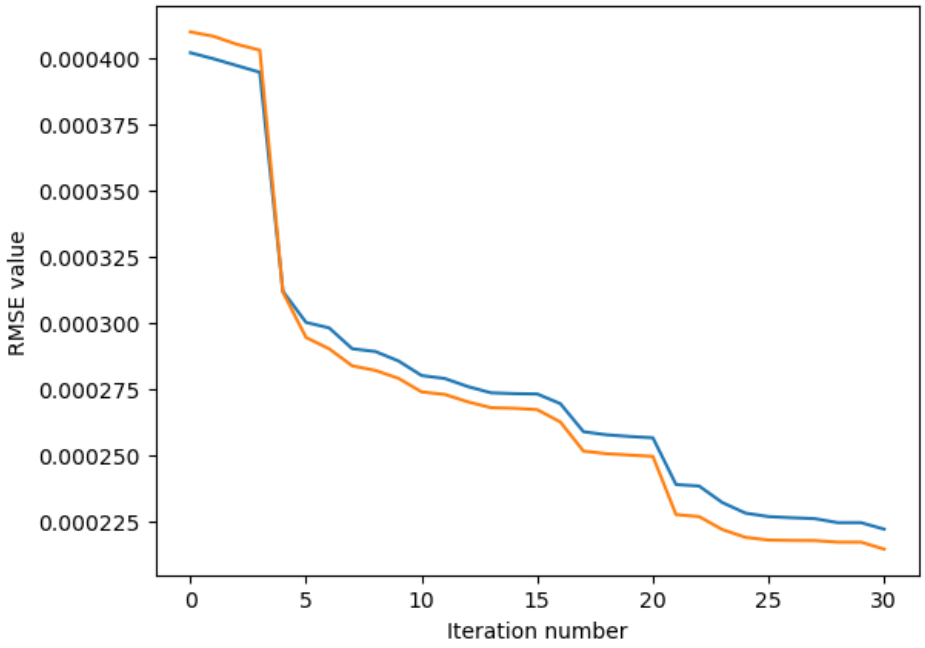
参数比较

比较不同超参数组合得到的模型性能

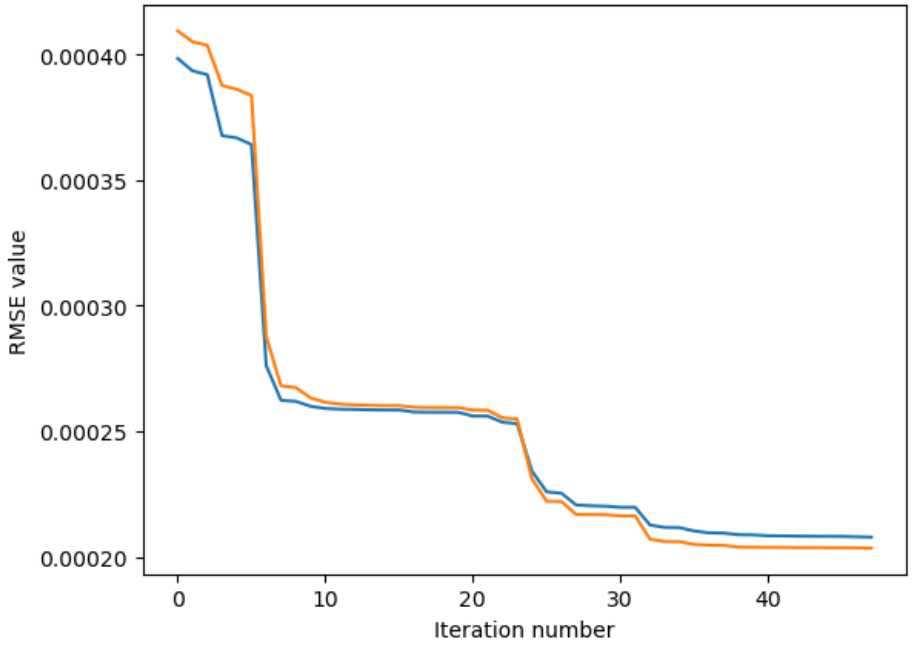
ESR=20 max_depth=50 learning_rate=1 gamma=1 Lambda=1 时模型的性能最佳

ESR	depth	lr	gamma	lambda	tr_MSRE	tr_R2	te_MSRE	te_R2
20	5	1	1	0.1	0.00023	0.6805	0.00026	0.5903
20	5	1	1	1	0.00029	0.5081	0.00031	0.4344
20	5	1	0.1	0.1	0.00026	0.5975	0.00029	0.4845
20	5	1	0.1	1	0.00019	0.7819	0.00024	0.6584
20	5	1	0.01	0.1	0.00028	0.5297	0.00031	0.4254
20	5	1	0.01	1	0.00031	0.4172	0.00033	0.3382
20	5	0.1	1	0.1	0.00035	0.2695	0.00042	-0.0451
20	5	0.1	1	1	0.00032	0.3797	0.00039	0.1083
20	5	0.1	0.1	0.1	0.00035	0.2775	0.00041	-0.0267
20	5	0.1	0.1	1	0.00036	0.2056	0.00044	-0.1371
20	5	0.1	0.01	0.1	0.00034	0.3112	0.00041	0.0014
20	5	0.1	0.01	1	0.00034	0.3062	0.00041	0.0062
20	50	1	1	0.1	0.00022	0.7148	0.00029	0.5049
20	**50**	**1**	**1**	**1**	**0.00016**	**0.8474**	**0.00022**	**0.6997**
20	50	1	0.1	0.1	0.00022	0.6982	0.00028	0.5160
20	50	1	0.1	1	0.00018	0.8130	0.00025	0.6227
20	50	1	0.01	0.1	0.00022	0.7044	0.00029	0.5035
20	50	1	0.01	1	0.00025	0.6065	0.00030	0.4637
20	50	0.1	1	0.1	0.00035	0.2670	0.00043	-0.1012
20	50	0.1	1	1	0.00033	0.3538	0.00040	0.1086
20	50	0.1	0.1	0.1	0.00031	0.4280	0.00039	0.1086
20	50	0.1	0.1	1	0.00032	0.3656	0.00040	0.0479
20	50	0.1	0.01	0.1	0.00033	0.3256	0.00042	-0.0395
20	50	0.1	0.01	1	0.00033	0.3471	0.00041	0.0007
20	500	1	1	0.1	0.00021	0.7249	0.00028	0.5141
20	500	1	1	1	0.00021	0.7357	0.00025	0.6111
20	500	1	0.1	0.1	0.00024	0.6544	0.00030	0.4511
20	500	1	0.1	1	0.00025	0.6307	0.00028	0.5235
20	500	1	0.01	0.1	0.00018	0.8028	0.00025	0.6326
20	500	1	0.01	1	0.00019	0.7751	0.00028	0.5227
20	500	0.1	1	0.1	0.00033	0.3412	0.00041	-0.0263
20	500	0.1	1	1	0.00033	0.3311	0.00041	0.0087
20	500	0.1	0.1	0.1	0.00035	0.2663	0.00043	-0.1018
20	500	0.1	0.1	1	0.00034	0.2855	0.00042	-0.0718
20	500	0.1	0.01	0.1	0.00035	0.2761	0.00043	-0.0965
20	500	0.1	0.01	1	0.00033	0.3262	0.00041	-0.0068

loss 曲线



此时发现 loss 曲线似乎并没有趋于稳定，故增大 epoch 至 500，重新训练得到的 loss 曲线为



***loss 曲线中的迭代次数为有效迭代次数，即 loss 减小时才将新的决策树加入模型 ***

实验总结

因为算法在选择特征时为随机选择，每个特征对模型的增益可能不同，故同参数多次训练产生的结果可能不同，减小这种差异的办法为：适当提高 epoch，使训练的停止策略为 2，即此时没有能使模型 loss 下降的特征。