作业 3：决策树与提升算法

2 决策树（40pt）

2.1 ID3 决策树算法（15pt）

1. 虽然表中没有列出数据集中的具体实例，但已经足以构建一棵 ID3 决策树的根节点（决策树桩）。请通过计算说明应该如何构建。

计算按是否晴天分类的信息增益

$$\begin{aligned}
H(\mathcal{D}) &= -\frac{35}{50}\log\frac{35}{50} - \frac{15}{50}\log\frac{15}{50} &\approx 0.8813 \\
H(\mathcal{D}_1) &= -\frac{28}{34}\log\frac{28}{34} - \frac{6}{34}\log\frac{6}{34} &\approx 0.6723 \\
H(\mathcal{D}_2) &= -\frac{7}{16}\log\frac{7}{16} - \frac{9}{16}\log\frac{9}{16} &\approx 0.9887
\end{aligned}$$

$$H(\mathcal{D}) - \frac{|\mathcal{D}_1|}{|\mathcal{D}|}H(\mathcal{D}_1) - \frac{|\mathcal{D}_2|}{|\mathcal{D}|}H(\mathcal{D}_2) = 0.8813 - \frac{34}{50}\times 0.6723 - \frac{16}{50}\times 0.9887 \approx 0.1077$$

计算按是否有雪分类的信息增益

$$\begin{aligned}
H(\mathcal{D}) &= -\frac{35}{50}\log\frac{35}{50} - \frac{15}{50}\log\frac{15}{50} &\approx 0.8813 \\
H(\mathcal{D}_1) &= -\frac{16}{18}\log\frac{16}{18} - \frac{2}{18}\log\frac{2}{18} &\approx 0.5033 \\
H(\mathcal{D}_2) &= -\frac{19}{32}\log\frac{19}{32} - \frac{13}{32}\log\frac{13}{32} &\approx 0.9745
\end{aligned}$$

$$H(\mathcal{D}) - \frac{|\mathcal{D}_1|}{|\mathcal{D}|}H(\mathcal{D}_1) - \frac{|\mathcal{D}_2|}{|\mathcal{D}|}H(\mathcal{D}_2) = 0.8813 - \frac{18}{50}\times 0.5033 - \frac{32}{50}\times 0.9745 \approx 0.0764$$

因此，根节点应该按是否晴天分类。

2. 使用最小错误替换 ID3 算法信息增益公式中的熵函数，构建一棵完整的决策树（展现过程）。

首先构造根节点。

$$\mathrm{MinError}(\mathcal{D}) = \min\left\{\frac{11}{24}, \frac{13}{24}\right\} = \frac{11}{24}$$

计算按颜色分类的信息增益

$$\begin{aligned}
\mathrm{MinError}(\mathcal{D}_1) &= \min\left\{\frac{5}{13}, \frac{8}{13}\right\} &= \frac{5}{13} \\
\mathrm{MinError}(\mathcal{D}_2) &= \min\left\{\frac{6}{11}, \frac{5}{11}\right\} &= \frac{5}{11}
\end{aligned}$$

$$\mathrm{MinError}(\mathcal{D}) - \frac{|\mathcal{D}_1|}{|\mathcal{D}|}\mathrm{MinError}(\mathcal{D}_1) - \frac{|\mathcal{D}_2|}{|\mathcal{D}|}\mathrm{MinError}(\mathcal{D}_2) = \frac{11}{24} - \frac{13}{24}\times\frac{5}{13} - \frac{11}{24}\times\frac{5}{11} = \frac{1}{24}$$

计算按大小分类的信息增益

$$\begin{aligned}
\mathrm{MinError}(\mathcal{D}_1) &= \min\left\{\frac{7}{18}, \frac{11}{18}\right\} &= \frac{7}{18} \\
\mathrm{MinError}(\mathcal{D}_2) &= \min\left\{\frac{2}{6}, \frac{4}{6}\right\} &= \frac{2}{6}
\end{aligned}$$

$$\text{MinError}\left(\mathcal{D}\right)-\frac{|\mathcal{D}_1|}{|\mathcal{D}|}\text{MinError}\left(\mathcal{D}_1\right)-\frac{|\mathcal{D}_2|}{|\mathcal{D}|}\text{MinError}\left(\mathcal{D}_2\right)=\frac{11}{24}-\frac{18}{24}\times\frac{7}{18}-\frac{6}{24}\times\frac{2}{6}=\frac{1}{12}$$

计算按动作分类的信息增益

$$\begin{aligned}\text{MinError}\left(\mathcal{D}_1\right)=\min\left\{\frac{5}{12},\frac{7}{12}\right\} &= \frac{5}{12}\\ \text{MinError}\left(\mathcal{D}_2\right)=\min\left\{\frac{6}{12},\frac{6}{12}\right\} &= \frac{6}{12}\end{aligned}$$

$$\text{MinError}\left(\mathcal{D}\right)-\frac{|\mathcal{D}_1|}{|\mathcal{D}|}\text{MinError}\left(\mathcal{D}_1\right)-\frac{|\mathcal{D}_2|}{|\mathcal{D}|}\text{MinError}\left(\mathcal{D}_2\right)=\frac{11}{24}-\frac{12}{24}\times\frac{5}{12}-\frac{12}{24}\times\frac{6}{12}=0$$

计算按年龄分类的信息增益

$$\begin{aligned}\text{MinError}\left(\mathcal{D}_1\right)=\min\left\{\frac{6}{12},\frac{6}{12}\right\} &= \frac{6}{12}\\ \text{MinError}\left(\mathcal{D}_2\right)=\min\left\{\frac{5}{12},\frac{7}{12}\right\} &= \frac{5}{12}\end{aligned}$$

$$\text{MinError}\left(\mathcal{D}\right)-\frac{|\mathcal{D}_1|}{|\mathcal{D}|}\text{MinError}\left(\mathcal{D}_1\right)-\frac{|\mathcal{D}_2|}{|\mathcal{D}|}\text{MinError}\left(\mathcal{D}_2\right)=\frac{11}{24}-\frac{12}{24}\times\frac{6}{12}-\frac{12}{24}\times\frac{5}{12}=0$$

因此，根节点应该按大小分类。

对于“小”的那部分，显然可以根据颜色或年龄分类。

对于“大”的那部分

$$\text{MinError}\left(\mathcal{D}\right)=\min\left\{\frac{7}{18},\frac{11}{18}\right\}=\frac{7}{18}$$

计算按颜色分类的信息增益

$$\begin{aligned}\text{MinError}\left(\mathcal{D}_1\right)=\min\left\{\frac{5}{13},\frac{8}{13}\right\} &= \frac{5}{13}\\ \text{MinError}\left(\mathcal{D}_2\right)=\min\left\{\frac{2}{5},\frac{3}{5}\right\} &= \frac{2}{5}\end{aligned}$$

$$\text{MinError}\left(\mathcal{D}\right)-\frac{|\mathcal{D}_1|}{|\mathcal{D}|}\text{MinError}\left(\mathcal{D}_1\right)-\frac{|\mathcal{D}_2|}{|\mathcal{D}|}\text{MinError}\left(\mathcal{D}_2\right)=\frac{7}{18}-\frac{13}{18}\times\frac{5}{13}-\frac{5}{18}\times\frac{2}{5}=0$$

计算按动作分类的信息增益

$$\begin{aligned}\text{MinError}\left(\mathcal{D}_1\right)=\min\left\{\frac{5}{12},\frac{7}{12}\right\} &= \frac{5}{12}\\ \text{MinError}\left(\mathcal{D}_2\right)=\min\left\{\frac{2}{6},\frac{4}{6}\right\} &= \frac{2}{6}\end{aligned}$$

$$\text{MinError}\left(\mathcal{D}\right)-\frac{|\mathcal{D}_1|}{|\mathcal{D}|}\text{MinError}\left(\mathcal{D}_1\right)-\frac{|\mathcal{D}_2|}{|\mathcal{D}|}\text{MinError}\left(\mathcal{D}_2\right)=\frac{7}{18}-\frac{12}{18}\times\frac{5}{12}-\frac{6}{18}\times\frac{2}{6}=0$$

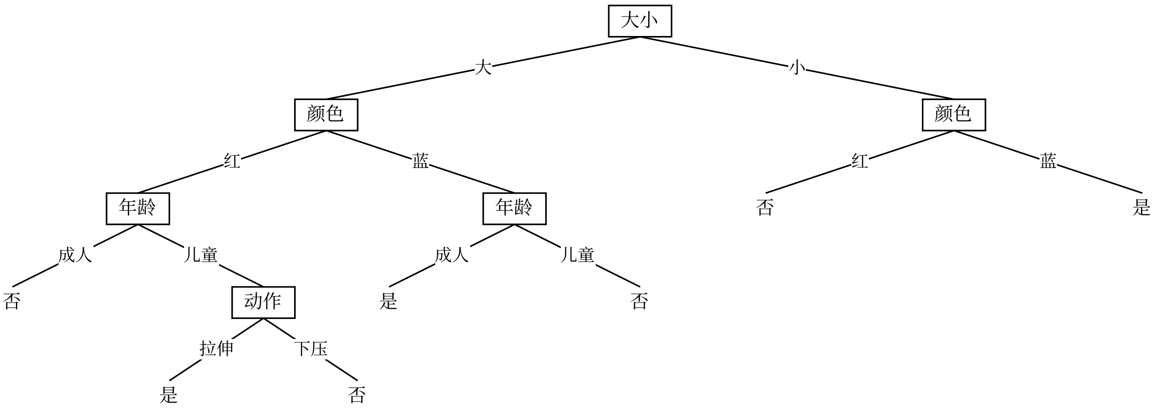
计算按年龄分类的信息增益

$$\begin{aligned}\text{MinError}\left(\mathcal{D}_1\right)=\min\left\{\frac{2}{8},\frac{6}{8}\right\} &= \frac{2}{8}\\ \text{MinError}\left(\mathcal{D}_2\right)=\min\left\{\frac{5}{10},\frac{5}{10}\right\} &= \frac{5}{10}\end{aligned}$$

$$\operatorname{MinError}(\mathcal{D})-\frac{|\mathcal{D}_1|}{|\mathcal{D}|}\operatorname{MinError}(\mathcal{D}_1)-\frac{|\mathcal{D}_2|}{|\mathcal{D}|}\operatorname{MinError}(\mathcal{D}_2)=\frac{7}{18}-\frac{8}{18}\times\frac{2}{8}-\frac{10}{18}\times\frac{5}{10}=0$$

由于三者信息增益相同，不妨选择颜色分类。

以此类推，可以构造出完整的决策树。



3. ID3 算法是否总能保证生成一颗“最优”的决策树？这里，“最优”的定义是：决策树能够最好的拟合训练数据，且深度最小。若能，请说明原因；若不能，请举出反例。

ID3 算法不能保证生成一颗全局最优的决策树，它是一种贪心算法，只是在局部选择最优属性分裂点，无法确保获得整体最优解。

我们可以考虑如下数据集

A	B	C	Label
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

在根节点处，由于选择 A、B、C 三个属性的信息增益相同，ID3 算法会随机选择一个属性进行分裂。假设选择 A 属性进行分裂，此时决策树深度为 3。事实上，如果选择 B 属性或 C 属性进行分裂，都可以得到深度为 2 的决策树。所以 ID3 算法不能保证生成一颗“最优”的决策树。

2.2 代码实验（25pt）

1. 根据熵的定义，完成 `tree.py` 中 `compute_entropy` 函数。

```python
def compute_entropy(label_array):
    """
    Calulate the entropy of given label list

    :param label_array: a numpy array of labels shape = (n, 1)
```

```python
    :return entropy: entropy value
    """
    # Your code goes here (~6 lines)
    # TODO 2.3.1
    _, label_counts = np.unique(label_array, return_counts=True)
    prob = label_counts / len(label_array)
    entropy = -np.sum(prob * np.log2(prob))
    return entropy
```

2. 根据基尼系数的定义，完成 `tree.py` 中 `compute_gini` 函数。

```python
def compute_gini(label_array):
    """
    Calulate the gini index of label list

    :param label_array: a numpy array of labels shape = (n, 1)
    :return gini: gini index value
    """
    # Your code goes here (~6 lines)
    # TODO 2.3.2
    _, label_counts = np.unique(label_array, return_counts=True)
    prob = label_counts / len(label_array)
    gini = 1 - np.sum(prob ** 2)
    return gini
```

3. 补全 `tree.py` 中 `DecisionTree` 类的 `fit` 函数。提示：递归调用决策树的构造与 `fit` 函数。

```python
def fit(self, X, y=None):
    """
    This should fit the tree classifier by setting the values self.is_leaf,
    self.split_id (the index of the feature we want ot split on, if we're splitting),
    self.split_value (the corresponding value of that feature where the split is),
    and self.leaf_value, which is the prediction value if the tree is a leaf node.  If we
    are splitting the node, we should also init self.left and self.right to be DecisionTree
    objects corresponding to the left and right subtrees. These subtrees should be fit on
    the data that fall to the left and right, respectively, of self.split_value.
    This is a recurisive tree building procedure.

    :param X: a numpy array of training data, shape = (n, m)
    :param y: a numpy array of labels, shape = (n, 1)

    :return self
    """
    # If depth is max depth turn into leaf
    if self.depth == self.max_depth:
        self.is_leaf = True
        self.leaf_value = self.leaf_value_estimator(y)
        return self

```

```python
    # If reach minimun sample size turn into leaf
    if len(y) <= self.min_sample:
        self.is_leaf = True
        self.leaf_value = self.leaf_value_estimator(y)
        return self

    # If not is_leaf, i.e in the node, we should create left and right subtree
    # But First we need to decide the self.split_id and self.split_value that minimize loss
    # Compare with constant prediction of all X
    best_split_value = None
    best_split_id = None
    best_loss = self.split_loss_function(y)
    best_left_X = None
    best_right_X = None
    best_left_y = None
    best_right_y = None
    # Concatenate y into X for sorting together
    X = np.concatenate([X, y], 1)
    for i in range(X.shape[1] - 1):
        # Note: The last column of X is y now
        X = np.array(sorted(X, key=lambda x: x[i]))
        for split_pos in range(len(X) - 1):
            # :split_pos+1 will include the split_pos data in left_X
            left_X = X[:split_pos + 1, :-1]
            right_X = X[split_pos + 1:, :-1]
            # you need left_y to be in (n,1) i.e (-1,1) dimension
            left_y = X[:split_pos + 1, -1].reshape(-1, 1)
            right_y = X[split_pos + 1:, -1].reshape(-1, 1)
            left_loss = len(left_y) * self.split_loss_function(left_y) / len(y)
            right_loss = len(right_y) * self.split_loss_function(right_y) / len(y)
            # If any choice of splitting feature and splitting position results in better loss
            # record following information and discard the old one
            if ((left_loss + right_loss) < best_loss):
                best_split_value = X[split_pos, i]
                best_split_id = i
                best_loss = left_loss + right_loss
                best_left_X = left_X
                best_right_X = right_X
                best_left_y = left_y
                best_right_y = right_y

    # Condition when you have a split position that results in better loss
    # Your code goes here (~10 lines)
    # TODO 2.3.3
    if best_split_id != None:
        # build child trees and set spliting info
        self.split_id = best_split_id
        self.split_value = best_split_value
        self.left = DecisionTree(self.split_loss_function, self.leaf_value_estimator, self.depth+1, self.min_sample, max_depth=self.max_depth)
```

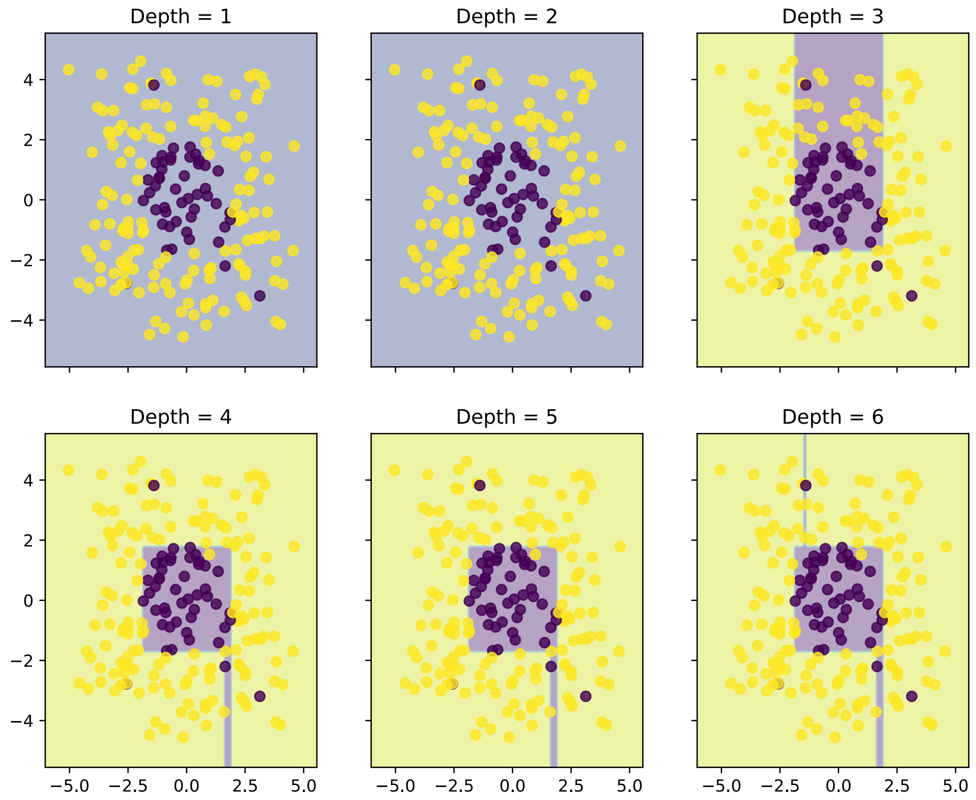
```
            self.right = DecisionTree(self.split_loss_function, self.leaf_value_estimator, self.depth+1, self.min_sample, max_depth=self.max_depth)
            self.left.fit(best_left_X, best_left_y)
            self.right.fit(best_right_X, best_right_y)
        else:
            # set value
            self.is_leaf = True
            self.leaf_value = self.leaf_value_estimator(y)

        return self
```

4. 完成 `tree.py` 中 `mean_absolute_deviation_around_median` 函数。

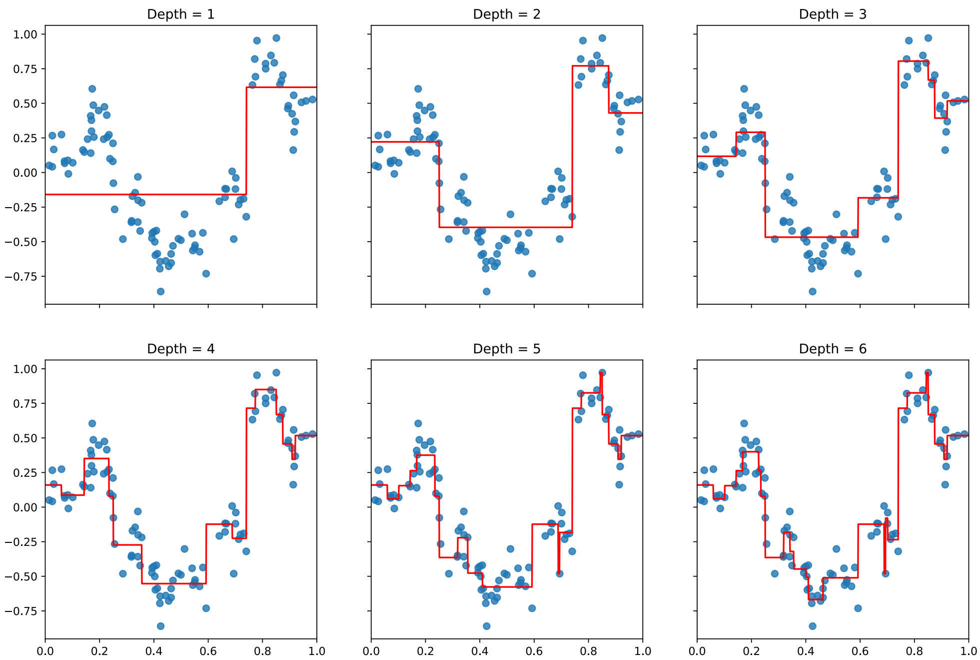
```
# Regression Tree Specific Code
def mean_absolute_deviation_around_median(y):
    """
    Calulate the mean absolute deviation around the median of a given target list

    :param y: a numpy array of targets shape = (n, 1)
    :return mae
    """
    # Your code goes here  (~3 lines)
    # TODO 2.3.4
    mae = np.mean(np.abs(y - np.median(y)))
    return mae
```

5. 运行 `tree.py`，在实验文档中记录决策树在不同数据集上运行的结果，包括 (a) `DT_entropy.pdf`，使用决策树在二分类问题上的结果。(b) `DT_regression.pdf`，使用决策树在回归问题上的结果。并简要描述实验现象（例如超参数对于决策树的影响）。

`DT_entropy.pdf` 的结果如下



`DT_regression.pdf` 的结果如下



可以观察到，在分类任务中，决策树的深度较小时（Depth = 1 或 Depth = 2），决策树完全无法进行分类；随着深度的增加，决策树的分类效果逐渐提升，但是深度过大时（Depth = 6）会出现过拟合现象。在回归任务中，决策树的深度较小时（Depth = 1），决策树的拟合效果较差，无法反映数据的变化；随着深度的增加，决策树对数据的拟合效果逐渐提升，但是深度过大时（Depth = 5 或 Depth = 6）会出现过拟合现象，即有很多小毛刺。

3 提升算法（60pt+10pt）

3.1 弱分类器的更新保证（10pt）

1. 通过计算证明： $Z_t \doteq \sum_{i=1}^{n} D_t(i) \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i)) = 2[\epsilon_t(1-\epsilon_t)]^{\frac{1}{2}}$ 。

归一化因子 Z_t 的定义为

$$Z_t = \sum_{i=1}^{n} D_t(i) \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))$$

其中，当分类正确，即 $y_i = h_t(\boldsymbol{x}_i)$ 时，有 $y_i h_t(\boldsymbol{x}_i) = 1$ ；当分类错误，即 $y_i \neq h_t(\boldsymbol{x}_i)$ 时，有 $y_i h_t(\boldsymbol{x}_i) = -1$ 。故而可以将 Z_t 分解为两部分：

$$Z_t = \sum_{y_i = h_t(\boldsymbol{x}_i)} D_t(i) \exp(-\alpha_t) + \sum_{y_i \neq h_t(\boldsymbol{x}_i)} D_t(i) \exp(\alpha_t)$$

由于有

$$\epsilon_t = \Pr_{i\sim D_t}[h_t(\boldsymbol{x}_i) \neq y_i] = \sum_{y_i \neq h_t(\boldsymbol{x}_i)} D_t(i)$$

故而可以将 Z_t 进一步写成

$$Z_t = \epsilon_t \exp(\alpha_t) + (1-\epsilon_t)\exp(-\alpha_t)$$

考虑到

$$\alpha_t \leftarrow \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

故而有

$$Z_t = \epsilon_t \exp\left(\frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)\right) + (1-\epsilon_t)\exp\left(-\frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)\right) = 2[\epsilon_t(1-\epsilon_t)]^{\frac{1}{2}}$$

2. 证明：h_t 关于分布 D_{t+1} 的错误率正好为 $\frac{1}{2}$ ，即对任意 $1 \leq t < T$：$\sum_{i=1}^{n} D_{t+1}(i)\mathbb{1}_{[y_i \neq h_t(\boldsymbol{x}_i)]} = \frac{1}{2}$ ，并据此说明，对任意 $1 \leq t < T$ ，$t+1$ 步选取的弱分类器 h_{t+1} 不会与 h_t 相同。

显然

$$\sum_{i=1}^{n} D_{t+1}(i)\mathbb{1}_{[y_i \neq h_t(\boldsymbol{x}_i)]} + \sum_{i=1}^{n} D_{t+1}(i)\mathbb{1}_{[y_i = h_t(\boldsymbol{x}_i)]} = 1$$

考虑到

$$D_{t+1}(i) \leftarrow \frac{D_t(i)\exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{Z_t}$$

其中，当分类正确，即 $y_i = h_t(\boldsymbol{x}_i)$ 时，有 $y_i h_t(\boldsymbol{x}_i) = 1$ ；当分类错误，即 $y_i \neq h_t(\boldsymbol{x}_i)$ 时，有 $y_i h_t(\boldsymbol{x}_i) = -1$ 。故而可以将上式改写成：

$$\sum_{y_i \neq h_t(\boldsymbol{x}_i)} \frac{D_t(i)\exp(\alpha_t)}{Z_t} + \sum_{y_i = h_t(\boldsymbol{x}_i)} \frac{D_t(i)\exp(-\alpha_t)}{Z_t} = 1$$

又因为

$$\epsilon_t = \Pr_{i\sim D_t}\left[h_t\left(\boldsymbol{x}_i\right) \neq y_i\right] = \sum_{y_i \neq h_t(\boldsymbol{x}_i)} D_t\left(i\right)$$

和

$$\alpha_t \leftarrow \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

以及上一题证明的

$$Z_t = 2\left[\epsilon_t\left(1-\epsilon_t\right)\right]^{\frac{1}{2}}$$

故而有

$$\sum_{y_i \neq h_t(\boldsymbol{x}_i)} \frac{D_t\left(i\right)\exp\left(\alpha_t\right)}{Z_t} = \sum_{y_i = h_t(\boldsymbol{x}_i)} \frac{D_t\left(i\right)\exp\left(-\alpha_t\right)}{Z_t} = \frac{1}{2}$$

即

$$\sum_{i=1}^{n} D_{t+1}\left(i\right)\mathbb{1}_{\left[y_i \neq h_t(\boldsymbol{x}_i)\right]} = \sum_{i=1}^{n} D_{t+1}\left(i\right)\mathbb{1}_{\left[y_i = h_t(\boldsymbol{x}_i)\right]} = \frac{1}{2}$$

3.2 替换目标函数（10pt）

1. 考虑如下的函数：(1) $\phi_1\left(x\right) = \mathbb{1}_{x\geq 0}$；(2) $\phi_2\left(x\right) = \left(1+x\right)^2$；(3) $\phi_3\left(x\right) = \max\left\{0, 1+x\right\}$；(4) $\phi_4\left(x\right) = \log_2\left(1+e^x\right)$。哪一个函数满足题面中对于 ϕ 的假设条件？请求出使用该函数时，$D_t\left(i\right)$ 的表达式。

显然只有 $\phi_4\left(x\right) = \log_2\left(1+e^x\right)$ 是单调递增且处处可微的凸函数，且满足 $\forall x \geq 0, \phi\left(x\right) \geq 1$ 且 $\forall x < 0, \phi\left(x\right) > 0$ 的条件。

对于 $\phi_4\left(x\right) = \log_2\left(1+e^x\right)$，有

$$\phi_4'\left(x\right) = \frac{1}{\ln 2}\frac{e^x}{1+e^x} = \frac{1}{\ln 2}\frac{1}{1+e^{-x}}$$

故

$$D_t\left(i\right) = \frac{\phi_4'\left(-y_i f_t\left(\boldsymbol{x}_i\right)\right)}{Z_t} = \frac{1}{Z_t \ln 2}\frac{1}{1+e^{y_i f_t(\boldsymbol{x}_i)}}$$

2. 上文的分析只确定了坐标下降中每次选择的最优坐标方向；对于前一问中选择的函数 ϕ，请通过求解 $\frac{\mathrm{d}L\left(\boldsymbol{\alpha}+\beta\boldsymbol{e}_t\right)}{\mathrm{d}\beta} = 0$，进一步确定最优步长 β。（写出 β 应满足的方程并适当化简即可，无需求解方程）

目标函数为

$$L\left(\boldsymbol{\alpha}\right) = \sum_{i=1}^{n} \log_2\left(1+e^{-y_i f(\boldsymbol{x}_i)}\right)$$

其中

$$f\left(\boldsymbol{x}_i\right) = \sum_{t=1}^{T} \alpha_t h_t\left(\boldsymbol{x}_i\right)$$

引入步长 β 后，目标函数变为

$$L\left(\boldsymbol{\alpha}+\beta \boldsymbol{e}_t\right)=\sum_{i=1}^n \log_2\left(1+e^{-y_i\left(f\left(\boldsymbol{x}_i\right)+\beta h_t\left(\boldsymbol{x}_i\right)\right)}\right)$$

对 β 求导，有

$$\frac{\mathrm{d} L\left(\boldsymbol{\alpha}+\beta \boldsymbol{e}_t\right)}{\mathrm{d} \beta}=\frac{1}{\ln 2} \sum_{i=1}^n-\frac{y_i h_t\left(\boldsymbol{x}_i\right) e^{-y_i\left(f\left(\boldsymbol{x}_i\right)+\beta h_t\left(\boldsymbol{x}_i\right)\right)}}{1+e^{-y_i\left(f\left(\boldsymbol{x}_i\right)+\beta h_t\left(\boldsymbol{x}_i\right)\right)}}$$

最优步长 β 应满足上式等于 0，适当化简后即为

$$\sum_{i=1}^n \frac{y_i h_t\left(\boldsymbol{x}_i\right)}{1+e^{y_i\left(f\left(\boldsymbol{x}_i\right)+\beta h_t\left(\boldsymbol{x}_i\right)\right)}}=0$$

3.3 带未知标签的 Boosting 算法（附加题，10pt）

1. 用 ϵ_t^s 和 α_t 表示 Z_t。

$$\begin{aligned}
Z_t & =\sum_{i=1}^n D_t(i) \exp \left(-\alpha_t y_i h_t\left(\boldsymbol{x}_i\right)\right) \\
& =\sum_{s=-1}^1 \sum_{i=1}^n D_t(i) \exp \left(-\alpha_t s\right) \mathbb{1}_{y_i h_t\left(\boldsymbol{x}_i\right)=s} \\
& =\sum_{s=-1}^1 \epsilon_t^s \exp \left(-\alpha_t s\right) \\
& =\epsilon_t^{-} e^{\alpha_t}+\epsilon_t^0+\epsilon_t^{+} e^{-\alpha_t}
\end{aligned}$$

2. 计算 $F'\left(\bar{\alpha}_{t-1}, \boldsymbol{e}_k\right)$，并指出第 t 步时弱分类器的优化目标。（结果用含 ϵ_t^s 的表达式表示）

由于

$$\begin{aligned}
F\left(\bar{\alpha}_{t-1}\right) & =\frac{1}{n} \sum_{i=1}^n e^{-y_i \sum_{j=1}^N \bar{\alpha}_{t-1, j} h_j\left(\boldsymbol{x}_i\right)} \\
F\left(\bar{\alpha}_{t-1}+\eta \boldsymbol{e}_k\right) & =\frac{1}{n} \sum_{i=1}^n e^{-y_i \sum_{j=1}^N \bar{\alpha}_{t-1, j} h_j\left(\boldsymbol{x}_i\right)-\eta y_i h_k\left(\boldsymbol{x}_i\right)}
\end{aligned}$$

故而有

$$\begin{aligned}
F'\left(\bar{\alpha}_{t-1}, \boldsymbol{e}_k\right) & =\lim_{\eta \rightarrow 0} \frac{F\left(\bar{\alpha}_{t-1}+\eta \boldsymbol{e}_k\right)-F\left(\bar{\alpha}_{t-1}\right)}{\eta} \\
& =-\frac{1}{n} \sum_{i=1}^n y_i h_k\left(\boldsymbol{x}_i\right) e^{-y_i \sum_{j=1}^N \bar{\alpha}_{t-1, j} h_j\left(\boldsymbol{x}_i\right)} \\
& =-\frac{1}{n} \sum_{i=1}^n y_i h_k\left(\boldsymbol{x}_i\right) \bar{D}_t(i) \bar{Z}_t \\
& =-\left[\sum_{i=1}^n \bar{D}_t(i) \mathbb{1}_{y_i h_t\left(\boldsymbol{x}_i\right)=1}-\sum_{i=1}^n \bar{D}_t(i) \mathbb{1}_{y_i h_t\left(\boldsymbol{x}_i\right)=-1}\right] \frac{\bar{Z}_t}{n} \\
& =-\left[\epsilon_t^{+}-\epsilon_t^{-}\right] \frac{\bar{Z}_t}{n}
\end{aligned}$$

第 t 步时弱分类器的优化目标为在分布 $\bar{D}_t$ 下，使得 $\left|\epsilon_t^{+}-\epsilon_t^{-}\right|$ 最小。

3. 解 $\dfrac{\partial F\left(\bar{\alpha}_{t-1}+\eta \boldsymbol{e}_k\right)}{\partial \eta}=0$，并给出 α_t 的更新公式。（结果用含 ϵ_t^s 的表达式表示）

$$
\begin{aligned}
\frac{\partial F\left(\bar{\alpha}_{t-1}+\eta \boldsymbol{e}_k\right)}{\partial \eta}&=0\\
-\frac{1}{n} \sum_{i=1}^n y_i h_k\left(\boldsymbol{x}_i\right) e^{-y_i \sum_{j=1}^N \bar{\alpha}_{t-1, j} h_j\left(\boldsymbol{x}_i\right)-\eta y_i h_k\left(\boldsymbol{x}_i\right)}&=0\\
\sum_{i=1}^n y_i h_k\left(\boldsymbol{x}_i\right) \bar{D}_t(i) \bar{Z}_t e^{-\eta y_i h_k\left(\boldsymbol{x}_i\right)}&=0\\
\sum_{i=1}^n \bar{D}_t(i) y_i h_k\left(\boldsymbol{x}_i\right) e^{-\eta y_i h_k\left(\boldsymbol{x}_i\right)}&=0\\
\sum_{i=1}^n \bar{D}_t(i) \mathbb{1}_{y_i h_t\left(\boldsymbol{x}_i\right)=1} e^{-\eta}-\sum_{i=1}^n \bar{D}_t(i) \mathbb{1}_{y_i h_t\left(\boldsymbol{x}_i\right)=-1} e^\eta&=0\\
\epsilon_t^{+} e^{-\eta}-\epsilon_t^{-} e^\eta&=0
\end{aligned}
$$

故可以解得

$$
\eta=\frac{1}{2} \log \frac{\epsilon_t^{+}}{\epsilon_t^{-}}
$$

故而有

$$
\alpha_t=\alpha_{t-1}+\frac{\boldsymbol{e}_k}{2} \log \frac{\epsilon_t^{+}}{\epsilon_t^{-}}
$$

4. 在 AdaBoost 中，我们证明了训练误差 $\hat{\mathcal{E}}=\dfrac{1}{n} \sum_{i=1}^n \mathbb{1}_{y_i f\left(\boldsymbol{x}_i\right)<0} \leq \prod_{t=1}^T Z_t \leq \prod_{t=1}^T 2 \sqrt{\epsilon_t\left(1-\epsilon_t\right)}$。类似地，请求出带未知标签的 Boosting 算法中，用含 ϵ_t^s 的表达式表示的训练误差上界，并证明若每个弱学习器的误差满足 $\dfrac{\epsilon_t^{+}-\epsilon_t^{-}}{\sqrt{1-\epsilon_t^0}} \geq \gamma>0$，则有：$\dfrac{1}{n} \sum_{i=1}^n \mathbb{1}_{y_i f\left(\boldsymbol{x}_i\right)<0} \leq \exp \left(-\dfrac{\gamma^2 T}{2}\right)$，其中，$T$ 是弱学习器的个数。

$$
\begin{aligned}
\hat{\mathcal{E}} & =\frac{1}{n} \sum_{i=1}^n \mathbb{1}_{y_i f\left(\boldsymbol{x}_i\right)<0} \\
& \leq \frac{1}{n} \sum_{i=1}^n e^{-y_i f\left(\boldsymbol{x}_i\right)} \\
& =\frac{1}{n} \sum_{i=1}^n\left(n \prod_{t=1}^T Z_t\right) D_{T+1}(i) \\
& =\prod_{t=1}^T Z_t
\end{aligned}
$$

上面已经证明 $Z_t=\epsilon_t^{-} e^{\alpha_t}+\epsilon_t^0+\epsilon_t^{+} e^{-\alpha_t}$，而通过 $\dfrac{\partial Z_t}{\partial \alpha_t}=\epsilon_t^{-} e^{\alpha_t}-\epsilon_t^{+} e^{-\alpha_t}=0$ 可以解得 $\alpha_t=\dfrac{1}{2} \log \dfrac{\epsilon_t^{+}}{\epsilon_t^{-}}$，故而有

$$
Z_t \leq \epsilon_t^0+2 \sqrt{\epsilon_t^{+} \epsilon_t^{-}}
$$

故而有

$$\hat{\mathcal{E}}=\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}_{y_i f(\boldsymbol{x}_i)<0}\leq\prod_{t=1}^{T}Z_t\leq\prod_{t=1}^{T}\left(\epsilon_t^0+2\sqrt{\epsilon_t^+\epsilon_t^-}\right)$$

注意到 $\epsilon_t^++\epsilon_t^-=1-\epsilon_t^0$，故 $Z_t\leq\epsilon_t^0+2\sqrt{\epsilon_t^+\epsilon_t^-}=\epsilon_t^0+\sqrt{\left(1-\epsilon_t^0\right)^2-\left(\epsilon_t^+-\epsilon_t^-\right)^2}$。

故当 $\frac{\epsilon_t^+-\epsilon_t^-}{\sqrt{1-\epsilon_t^0}}\geq\gamma>0$ 即 $\left(\epsilon_t^+-\epsilon_t^-\right)^2\geq\gamma^2\left(1-\epsilon_t^0\right)$ 时，有

$$\begin{aligned}
\hat{\mathcal{E}}&=\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}_{y_i f(\boldsymbol{x}_i)<0}\\
&\leq\prod_{t=1}^{T}\left(\epsilon_t^0+\sqrt{\left(1-\epsilon_t^0\right)^2-\left(\epsilon_t^+-\epsilon_t^-\right)^2}\right)\\
&\leq\prod_{t=1}^{T}\left(\epsilon_t^0+\sqrt{\left(1-\epsilon_t^0\right)^2-\gamma^2\left(1-\epsilon_t^0\right)}\right)\\
&\leq\prod_{t=1}^{T}\left(\epsilon_t^0+\left(1-\epsilon_t^0\right)\sqrt{1-\gamma^2}\right)\\
&\leq\prod_{t=1}^{T}\sqrt{\epsilon_t^0+\left(1-\epsilon_t^0\right)\left(1-\gamma^2\right)}\\
&=\prod_{t=1}^{T}\sqrt{1-\left(1-\epsilon_t^0\right)\gamma^2}\\
&\leq\prod_{t=1}^{T}\sqrt{\exp\left(-\left(1-\epsilon_t^0\right)\gamma^2\right)}\\
&\leq\prod_{t=1}^{T}\sqrt{\exp\left(-\gamma^2\right)}\\
&=\exp\left(-\frac{\gamma^2T}{2}\right)
\end{aligned}$$

3.4 Gradient Boosting Machines（40pt）

1. 完成上述算法中的填空。

> 1. 令 $f_0(\boldsymbol{x})=0$。
> 2. For $t=1$ to T：
>
>   (a) 计算在各个数据点上的梯度 $\boldsymbol{g}_t=\left(\frac{\partial}{\partial\hat{\boldsymbol{y}}_i}\ell\left(\boldsymbol{y}_i,\hat{\boldsymbol{y}}_i\right)\Big|_{\hat{\boldsymbol{y}}_i=f_{t-1}(\boldsymbol{x}_i)}\right)_{i=1}^{n}$。
>
>   (b) 根据 $-\boldsymbol{g}_t$ 拟合一个回归模型， $h_t=\arg\min_{h\in\mathcal{F}}\sum_{i=1}^{n}\left(h\left(x_i\right)-\boldsymbol{g}_t\right)^2$。
>
>   (c) 选择合适的步长 α_t, 最简单的选择是固定步长 $\eta\in(0,1]$。
>
>   (d) 更新模型， $f(\boldsymbol{x})=f_{t-1}(\boldsymbol{x})+\eta h_t(\boldsymbol{x})$。

2. 考虑回归问题，假设损失函数 $\ell(\boldsymbol{y},\hat{\boldsymbol{y}})=\frac{1}{2}(\boldsymbol{y}-\hat{\boldsymbol{y}})^2$。直接给出第 t 轮迭代时的 $\boldsymbol{g}_t$ 以及 h_t 的表达式。（使用 f_{t-1} 表达）。

$$
\begin{aligned}
\boldsymbol{g}_t &= \frac{\partial}{\partial \hat{\boldsymbol{y}}} \frac{1}{2}(\boldsymbol{y}-\hat{\boldsymbol{y}})^2 \bigg|_{\hat{\boldsymbol{y}}=f_{t-1}(\boldsymbol{x})} \\
&= f_{t-1}(\boldsymbol{x}) - \boldsymbol{y}
\end{aligned}
$$

$$
\begin{aligned}
h_t &= \arg\min_{h\in\mathcal{F}} \sum_{i=1}^{n} \left(h(\boldsymbol{x}_i) - \boldsymbol{g}_t\right)^2 \\
&= \arg\min_{h\in\mathcal{F}} \sum_{i=1}^{n} \left(h(\boldsymbol{x}_i) - f_{t-1}(\boldsymbol{x}_i) + \boldsymbol{y}\right)^2
\end{aligned}
$$

3. 考虑二分类问题，假设损失函数 $\ell(\boldsymbol{y},\hat{\boldsymbol{y}})=\ln\left(1+e^{-\boldsymbol{y}\hat{\boldsymbol{y}}}\right)$。直接给出第 t 轮迭代时的 $\boldsymbol{g}_t$ 以及 h_t 的表达式。（使用 f_{t-1} 表达）。

$$
\begin{aligned}
\boldsymbol{g}_t &= \frac{\partial}{\partial \hat{\boldsymbol{y}}} \ln\left(1+e^{-\boldsymbol{y}\hat{\boldsymbol{y}}}\right) \bigg|_{\hat{\boldsymbol{y}}=f_{t-1}(\boldsymbol{x})} \\
&= \frac{-\boldsymbol{y}e^{-\boldsymbol{y}f_{t-1}(\boldsymbol{x})}}{1+e^{-\boldsymbol{y}f_{t-1}(\boldsymbol{x})}} \\
&= -\boldsymbol{y}\sigma\left(-\boldsymbol{y}f_{t-1}(\boldsymbol{x})\right)
\end{aligned}
$$

$$
\begin{aligned}
h_t &= \arg\min_{h\in\mathcal{F}} \sum_{i=1}^{n} \left(h(\boldsymbol{x}_i) - \boldsymbol{g}_t\right)^2 \\
&= \arg\min_{h\in\mathcal{F}} \sum_{i=1}^{n} \left(h(\boldsymbol{x}_i) + \boldsymbol{y}\sigma\left(-\boldsymbol{y}f_{t-1}(\boldsymbol{x}_i)\right)\right)^2
\end{aligned}
$$

4. 完成 `boosting.py` 中 `GradientBoosting` 类的 `fit` 函数。

```python
def fit(self, train_data, train_target):
    """
    Fit gradient boosting model
    :param train_data: x
    :param train_target: y
    :return:
    """
    ft = np.zeros(train_data.shape[0])  # f_t(x)
    train_target = train_target.squeeze()
    self.ht = []  # sequence of h_t(x)
    for t in range(self.T):
        rgs = DecisionTreeRegressor(min_samples_split=self.min_sample,
max_depth=self.max_depth)
        # Your code goes here (~4 lines)
        # TODO 3.5.4
        gradient = self.gradient_func(train_target, ft)
        rgs.fit(train_data, -gradient)
        self.ht.append(rgs)
        ft += self.learning_rate * rgs.predict(train_data)

    return self
```

5. 完成 `boosting.py` 中 `GradientBoosting` 类的 `predict` 函数。

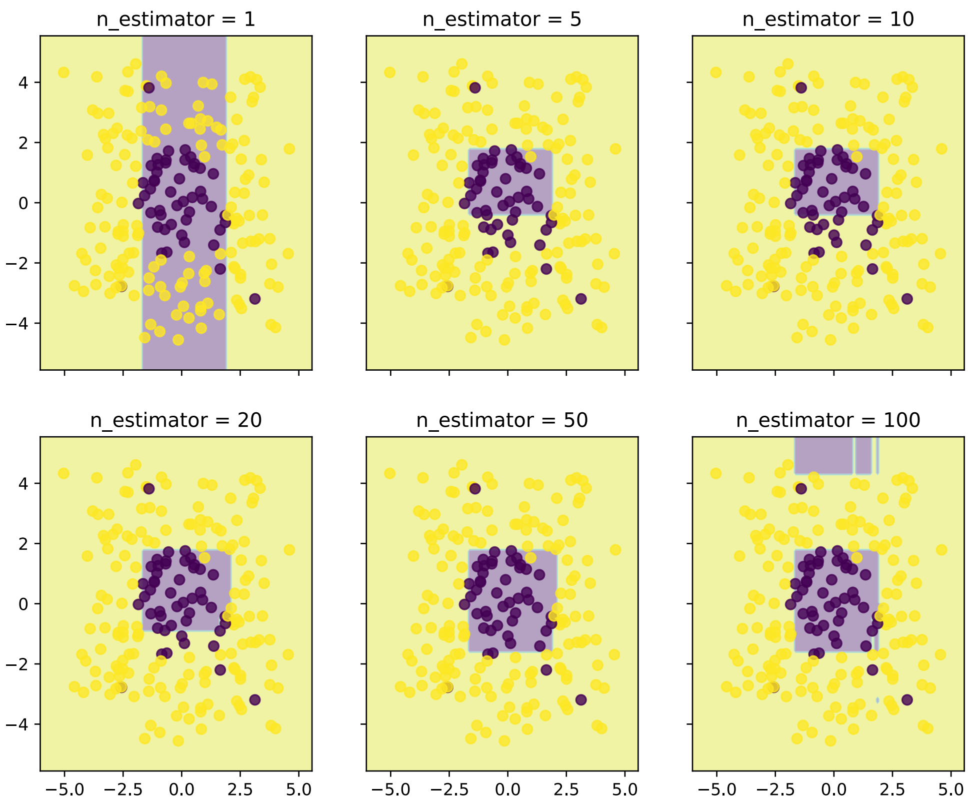
```
def predict(self, test_data):
    # Your code goes here (~6 lines)
    # TODO 3.5.5
    ft = np.zeros(test_data.shape[0])
    for rgs in self.ht:
        ft += self.learning_rate * rgs.predict(test_data)
    return ft
```

6. 完成 `boosting.py` 中函数 `gradient_logistic`。

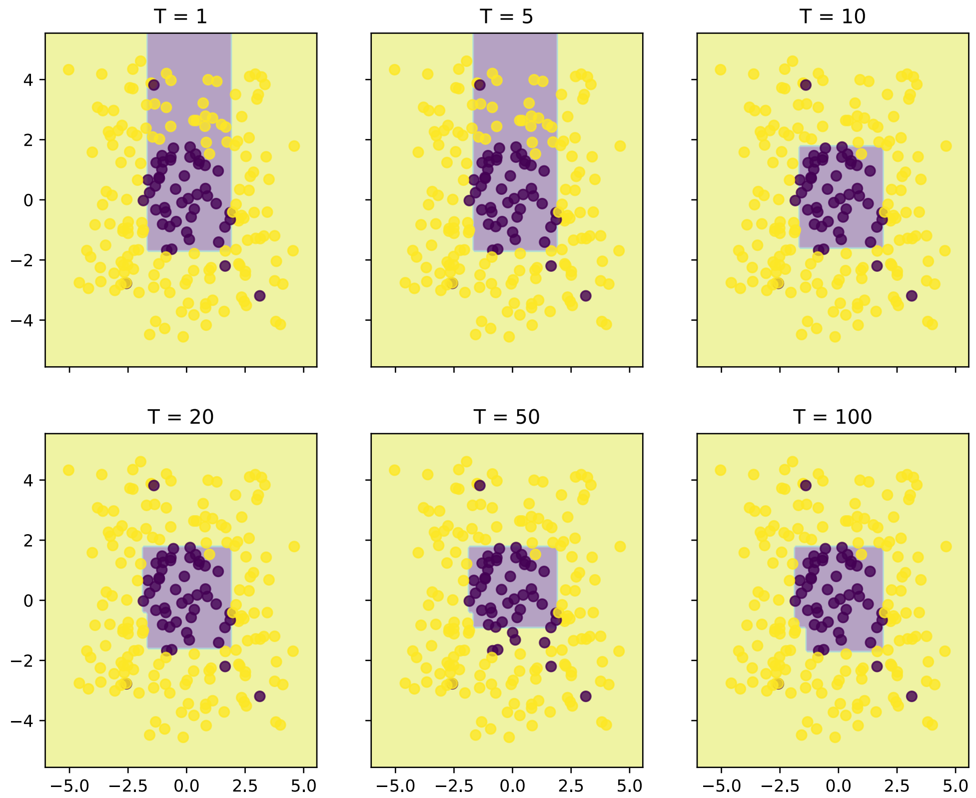
```
def gradient_logistic(train_target, train_predict):
    """
    compute g_t in 3.5.3
    """
    # Your code goes here  (~3 lines)
    # TODO 3.5.6
    sigmoid = 1 / (1 + np.exp(train_target * train_predict))
    return -train_target * sigmoid
```

7. 运行 `boosting.py`，在实验文档中记录 GBM 在不同数据集上运行的结果，包括 (a) `GBM_l2.pdf`，使用 L2 loss 在二分类问题上的结果。(b) `GBM_logistic.pdf`，使用 logistic loss 在二分类问题上的结果。(c) `GBM_regression.pdf`，使用 L2 loss 在回归问题上的结果。并简要描述实验现象（例如超参数对于 GBM 的影响、损失函数对于 GBM 的影响等）。

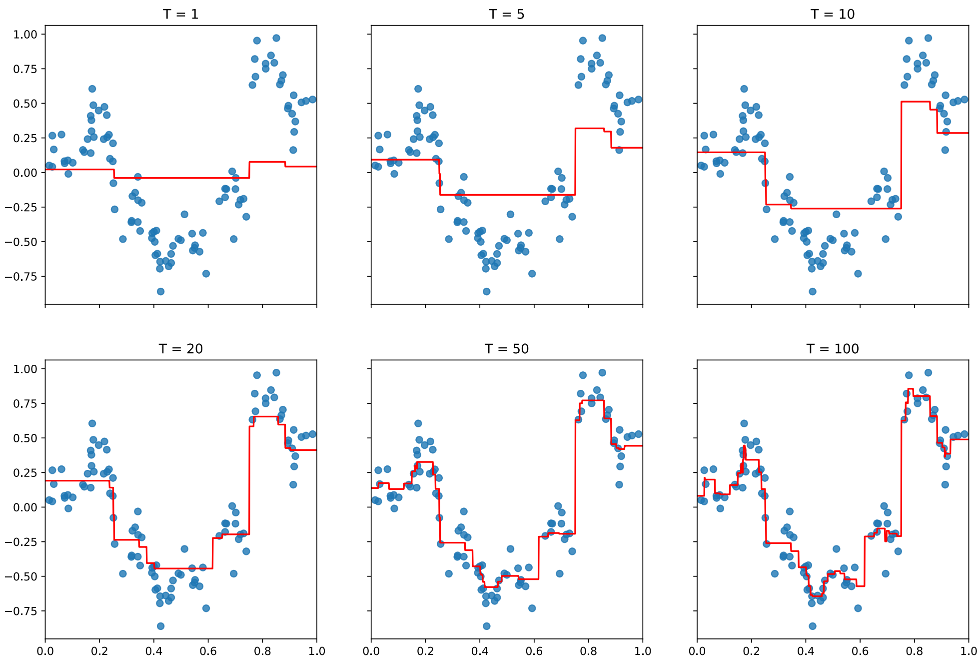
`GBM_l2.pdf` 的结果如下



`GBM_logistic.pdf` 的结果如下



`GBM_regression.pdf` 的结果如下



可以观察到，对于 L2 loss 在二分类问题上的结果，当 `n_estimator` 较小时，存在欠拟合现象；随着 `n_estimator` 的增加，模型的拟合效果逐渐提升，但是当 `n_estimator` 较大时，存在过拟合现象。

对于 logistic loss 在二分类问题上的结果，当 `T` 较小时，存在欠拟合现象；随着 `T` 的增加，模型的拟合效果逐渐提升，且整体上优于 L2 loss。

对于 L2 loss 在回归问题上的结果，当 `T` 较小时，存在欠拟合现象；随着 `T` 的增加，模型的拟合效果逐渐提升，但是当 `T` 较大时，存在少量的过拟合现象，但整体效果上优于决策树的结果。

成绩：100

评语：2.1.2 -3; 3.3（附加题） 10