THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 12**

Ensemble

Lecturer: Changshui Zhang `zcs@mail.tsinghua.edu.cn`

Hong Zhao `vzhao@tsinghua.edu.cn`

Student: Jingxuan Yang `yangjx20@mails.tsinghua.edu.cn`

Bagging

In practice, we have only a single data set, and *bagging* is a method to introduce variability between different models within the committee based on one data set. A committee can be viewed as a set of individual models on which we average our predictions.

The very first step is to use *bootstrap* data sets. After we have generated M bootstrap data sets, we then use each to train a separate predictive model y_m where $m = 1, 2, \dots, M$. Then the prediction is given by

$$y_{\text{COM}} = \frac{1}{M}\sum_{m=1}^{M} y_m(\boldsymbol{x}). \tag{1}$$

Suppose the true regression function that we are trying to predict is given by $h(\boldsymbol{x})$, so that the output of each of the models can be written as the true value plus an error in the form

$$y_m(\boldsymbol{x}) = h(\boldsymbol{x}) + \epsilon_m(\boldsymbol{x}). \tag{2}$$

The average sum-of-square error then takes the form

$$\mathbb{E}_{\boldsymbol{x}}[\{y_m(\boldsymbol{x}) - h(\boldsymbol{x})\}^2] = \mathbb{E}_{\boldsymbol{x}}[\epsilon_m^2(\boldsymbol{x})], \tag{3}$$

where $\mathbb{E}_{\boldsymbol{x}}$ denotes expectation with respect to the distribution of the input vector $\boldsymbol{x}$.

The average error made by the models acting individually is therefore

$$E_{\text{AV}} = \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{x}}[\epsilon_m^2(\boldsymbol{x})]. \tag{4}$$

Similarly, the expected error from equation (1) is given by

$$\begin{aligned} E_{\mathrm{COM}} &= \mathbb{E}_{\boldsymbol{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} y_m(\boldsymbol{x}) - h(\boldsymbol{x})\right\}^2\right] \\ &= \mathbb{E}_{\boldsymbol{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(\boldsymbol{x})\right\}^2\right]. \end{aligned} \tag{5}$$

1.1 Assume that errors have zero mean and are uncorrelated

$$\begin{gathered} \mathbb{E}_{\boldsymbol{x}}[\epsilon_m(\boldsymbol{x})] = 0, \\ \mathbb{E}_{\boldsymbol{x}}[\epsilon_m(\boldsymbol{x})\epsilon_l(\boldsymbol{x})] = 0,\ \forall\, m \neq l. \end{gathered} \tag{6}$$

Please prove that

$$E_{\mathrm{COM}} = \frac{1}{M} E_{\mathrm{AV}}. \tag{7}$$

解: 由误差不相关可得

$$\begin{aligned} E_{\mathrm{COM}} &= \mathbb{E}_{\boldsymbol{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(\boldsymbol{x})\right\}^2\right] \\ &= \frac{1}{M^2}\mathbb{E}_{\boldsymbol{x}}\left[\left\{\sum_{m=1}^{M} \epsilon_m(\boldsymbol{x})\right\}^2\right] \\ &= \frac{1}{M^2}\mathbb{E}_{\boldsymbol{x}}\left[\sum_{m=1}^{M}\sum_{l=1}^{M} \epsilon_m(\boldsymbol{x})\epsilon_l(\boldsymbol{x})\right] \\ &= \frac{1}{M^2}\mathbb{E}_{\boldsymbol{x}}\left[\sum_{m=1}^{M} \epsilon_m^2(\boldsymbol{x})\right] \\ &= \frac{1}{M^2}\sum_{m=1}^{M}\mathbb{E}_{\boldsymbol{x}}[\epsilon_m^2(\boldsymbol{x})] \\ &= \frac{1}{M}E_{\mathrm{AV}} \end{aligned} \tag{8}$$

1.2 In practice, the errors are typically highly correlated. Show that the following inequality holds without assumptions in 1.1.

$$E_{\mathrm{COM}} \leqslant E_{\mathrm{AV}}. \tag{9}$$

解: 函数 $f(x) = x^2$ 是凸函数, 则由 Jensen 不等式可得

$$\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(\boldsymbol{x})\right\}^2 \leqslant \frac{1}{M}\sum_{m=1}^{M} \epsilon_m^2(\boldsymbol{x}) \tag{10}$$

所以,

$$\begin{aligned} E_{\mathrm{COM}} &= \mathbb{E}_{\boldsymbol{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M}\epsilon_m(\boldsymbol{x})\right\}^2\right] \\ &\leqslant \mathbb{E}_{\boldsymbol{x}}\left[\frac{1}{M}\sum_{m=1}^{M}\epsilon_m^2(\boldsymbol{x})\right] \\ &= \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_{\boldsymbol{x}}[\epsilon_m^2(\boldsymbol{x})] \\ &= E_{\mathrm{AV}} \end{aligned} \tag{11}$$

即 $E_{\mathrm{COM}} \leqslant E_{\mathrm{AV}}$.

1.3 In the previous problem, our error function is $f(y(\boldsymbol{x})-h(\boldsymbol{x})) = (y(\boldsymbol{x})-h(\boldsymbol{x}))^2$ (sum-of-square). By making use of *Jensen's inequality*, show that equation (9) holds for any error function $E(y(\boldsymbol{x})-h(\boldsymbol{x}))$ provided it is a convex function of $y(\boldsymbol{x})-h(\boldsymbol{x})$.

解: 函数 $E(\cdot)$ 是凸函数, 则由 Jensen 不等式可得

$$E\left\{\frac{1}{M}\sum_{m=1}^{M}\epsilon_m(\boldsymbol{x})\right\} \leqslant \frac{1}{M}\sum_{m=1}^{M}E(\epsilon_m(\boldsymbol{x})) \tag{12}$$

所以,

$$\begin{aligned} E_{\mathrm{COM}} &= \mathbb{E}_{\boldsymbol{x}}\left[E\left\{\frac{1}{M}\sum_{m=1}^{M}\epsilon_m(\boldsymbol{x})\right\}\right] \\ &\leqslant \mathbb{E}_{\boldsymbol{x}}\left[\frac{1}{M}\sum_{m=1}^{M}E(\epsilon_m(\boldsymbol{x}))\right] \\ &= \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_{\boldsymbol{x}}[E(\epsilon_m(\boldsymbol{x}))] \\ &= E_{\mathrm{AV}} \end{aligned} \tag{13}$$

即 $E_{\mathrm{COM}} \leqslant E_{\mathrm{AV}}$.

1.4 Consider the case in which we allow unequal weighting of the individual models

$$y_{\mathrm{COM}}(\boldsymbol{x}) = \sum_{m=1}^{M}\alpha_m y_m(\boldsymbol{x}). \tag{14}$$

In order to make $y_{\mathrm{COM}}(\boldsymbol{x})$ sensible, we require that for all $y_m(\boldsymbol{x})$ they are bounded at each value of $\boldsymbol{x}$ like

$$y_{\min}(\boldsymbol{x}) \leqslant y_{\mathrm{COM}}(\boldsymbol{x}) \leqslant y_{\max}(\boldsymbol{x}). \tag{15}$$

Show that the necessary and sufficient condition for constraint (15) is

$$\alpha_m \geqslant 0, \quad \sum_{m=1}^{M} \alpha_m = 1. \tag{16}$$

解: 首先证明充分性, 令

$$\alpha_m \geqslant 0, \quad \sum_{m=1}^{M} \alpha_m = 1 \tag{17}$$

则有

$$y_{\mathrm{COM}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m y_m(\boldsymbol{x}) \geqslant \sum_{m=1}^{M} \alpha_m y_{\min}(\boldsymbol{x}) = y_{\min}(\boldsymbol{x}) \tag{18}$$

且

$$y_{\mathrm{COM}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m y_m(\boldsymbol{x}) \leqslant \sum_{m=1}^{M} \alpha_m y_{\max}(\boldsymbol{x}) = y_{\max}(\boldsymbol{x}) \tag{19}$$

下面证明必要性, 使用反证法. 假设

$$\sum_{m=1}^{M} \alpha_m \neq 1 \tag{20}$$

取

$$y_m(\boldsymbol{x}) = \tilde{y}(\boldsymbol{x}), \; \forall \, m = 1, 2, \ldots, M \tag{21}$$

则 $y_{\min}(\boldsymbol{x}) = y_{\max}(\boldsymbol{x}) = \tilde{y}(\boldsymbol{x})$, 而

$$y_{\mathrm{COM}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m y_m(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m \tilde{y}(\boldsymbol{x}) \neq \tilde{y}(\boldsymbol{x}) \tag{22}$$

与 $y_{\min}(\boldsymbol{x}) \leqslant y_{\mathrm{COM}}(\boldsymbol{x}) \leqslant y_{\max}(\boldsymbol{x})$ 矛盾, 故

$$\sum_{m=1}^{M} \alpha_m = 1 \tag{23}$$

假设 $\exists \, l \in \{1, 2, \ldots, M\}$ 使得 $\alpha_l < 0$, 则

$$\sum_{m \neq l} \alpha_m = 1 - \alpha_l > 1 \tag{24}$$

取 $y_l(\boldsymbol{x}) = 0$,

$$y_m(\boldsymbol{x}) = \tilde{y}(\boldsymbol{x}), \; \forall \, m \neq l \tag{25}$$

令

$$P \triangleq \{\boldsymbol{x} : \tilde{y}(\boldsymbol{x}) > 0\}, \quad N \triangleq \{\boldsymbol{x} : \tilde{y}(\boldsymbol{x}) < 0\} \tag{26}$$

则

$$y_{\max}(\boldsymbol{x}) = \max(0, \tilde{y}(\boldsymbol{x})) = \begin{cases} \tilde{y}(\boldsymbol{x}), & \text{if } \boldsymbol{x} \in P \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

$$y_{\min}(\boldsymbol{x}) = \min(0, \tilde{y}(\boldsymbol{x})) = \begin{cases} \tilde{y}(\boldsymbol{x}), & \text{if } \boldsymbol{x} \in N \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

对 $\forall\, \boldsymbol{x} \in P$, 由式 (24) 可得

$$y_{\text{COM}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m y_m(\boldsymbol{x}) = \sum_{m \neq l} \alpha_m \tilde{y}(\boldsymbol{x}) > \tilde{y}(\boldsymbol{x}) = y_{\max}(\boldsymbol{x}) \tag{29}$$

与 $y_{\text{COM}}(\boldsymbol{x}) \leqslant y_{\max}(\boldsymbol{x})$ 矛盾.

对 $\forall\, \boldsymbol{x} \in N$, 由式 (24) 可得

$$y_{\text{COM}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m y_m(\boldsymbol{x}) = \sum_{m \neq l} \alpha_m \tilde{y}(\boldsymbol{x}) < \tilde{y}(\boldsymbol{x}) = y_{\min}(\boldsymbol{x}) \tag{30}$$

与 $y_{\text{COM}}(\boldsymbol{x}) \geqslant y_{\min}(\boldsymbol{x})$ 矛盾.

由以上矛盾可知

$$\alpha_m \geqslant 0,\ \forall\, m = 1, 2, \ldots, M \tag{31}$$

Gradient Boosting

Gradient boosting is a generation of boosting algorithms, using the connection between boosting and optimization. For the boosting part, it builds an additive model in a forward stage-wise fashion. For the optimization part, it allows for the optimization of arbitrary differentiable loss functions by using their gradients.

In any function estimation problem, we wish to find a regression function $f(x) \in \mathcal{F}$ that minimizes the expectation of some loss function, where $f(x)$ is a function that maps from the input space to $\mathbb{R}$, and $\mathcal{F}$ is the hypothesis space of all possible regression functions.

Denote a given loss function as ℓ. The Gradient Boosting algorithm contains M steps. At each step, it tries to build a regression functions $h_m(x)$ and adds it to the ensembled function $f_m(x)$ to minimize ℓ. In the end all functions of M steps add up to form the final regression function $f_M(x)$. The details are described as follows.

1. Initialize $f_0(x) = 0$.

2. For $m = 1$ to M:

(a) Compute the gradient:

$$(\boldsymbol{g}_m)_i = \frac{\partial}{\partial f(x_i)} \ell(y_i, f(x_i)) \Big|_{f(x_i)=f_{m-1}(x_i)}, \tag{32}$$

where $\{y_i, x_i\}_1^n$ are n data samples.

(b) The negative gradient $-\boldsymbol{g}_m$ is said to define the "steepest-descent" direction. Thus, we could use the negative gradient as the working response and fit regression model to $-\boldsymbol{g}_m$:

$$h_m = \operatorname*{argmin}_{h \in \mathcal{F}} \sum_{i=1}^{n} \left((-\boldsymbol{g}_m)_i - h(x_i)\right)^2, \tag{33}$$

each $h_m \in \mathcal{F}$ is chosen in a learning process.

(c) Choose fixed step size $\nu_m = \nu \in (0, 1]$, or take

$$\nu_m = \operatorname*{argmin}_{\nu > 0} \sum_{i=1}^{n} \ell\left(y_i, f_{m-1}(x_i) + \nu h_m(x_i)\right), \tag{34}$$

where ν_m is the size of the step along the direction of the greatest descent.

(d) Update the estimate of $f(x)$ as:

$$f_m(x) = f_{m-1}(x) + \nu_m h_m(x). \tag{35}$$

3. Return f_M.

In this problem we'll derive two special cases of the general gradient boosting framework: L_2-Boosting and Binomial Boost.

2.1 Consider the regression framework, where label space $\mathcal{Y} = \mathbb{R}$. Suppose our loss function is given by

$$\ell(\hat{y}, y) = \frac{1}{2}\left(\hat{y} - y\right)^2, \tag{36}$$

and at the beginning of the m'th round of gradient boosting, we have the function $f_{m-1}(x)$. Show that the h_m chosen as the next basis function is given by

$$h_m = \operatorname*{argmin}_{h \in \mathcal{F}} \sum_{i=1}^{n} \left[(y_i - f_{m-1}(x_i)) - h(x_i)\right]^2. \tag{37}$$

In other words, at each stage we find the weak prediction function $h_m \in \mathcal{F}$ that is the best fit to the residuals from the previous stage.

Hint: Once you understand what's going on, this is a pretty easy problem.

解: 由损失函数的定义可得

$$\ell(y_i, f(x_i)) = \frac{1}{2}(y_i - f(x_i))^2 \tag{38}$$

则

$$\begin{aligned}(\boldsymbol{g}_m)_i &= \frac{\partial}{\partial f(x_i)}\ell(y_i, f(x_i))\bigg|_{f(x_i)=f_{m-1}(x_i)} \\ &= \frac{\partial}{\partial f_{m-1}(x_i)}\left[\frac{1}{2}(y_i - f_{m-1}(x_i))^2\right] \\ &= -(y_i - f_{m-1}(x_i))\end{aligned} \tag{39}$$

所以,

$$\begin{aligned}h_m &= \operatorname*{argmin}_{h\in\mathcal{F}}\sum_{i=1}^{n}\left[(-\boldsymbol{g}_m)_i - h(x_i)\right]^2 \\ &= \operatorname*{argmin}_{h\in\mathcal{F}}\sum_{i=1}^{n}\left[(y_i - f_{m-1}(x_i)) - h(x_i)\right]^2\end{aligned} \tag{40}$$

2.2 Now let's consider the classification framework, where $\mathcal{Y} = \{-1, 1\}$. This time, let's consider the logistic loss

$$\ell(m) = \ln\left(1 + e^{-m}\right), \tag{41}$$

where $m = yf(x)$ is the margin. Similar to what we did in the L_2-Boosting question, write an expression for h_m as an argmin over $\mathcal{F}$.

解: 由损失函数的定义可得

$$\ell(y_i, f(x_i)) = \ln[1 + \exp(-y_i f(x_i))] \tag{42}$$

则

$$\begin{aligned}(\boldsymbol{g}_m)_i &= \frac{\partial}{\partial f(x_i)}\ell(y_i, f(x_i))\bigg|_{f(x_i)=f_{m-1}(x_i)} \\ &= \frac{\partial \ln[1 + \exp(-y_i f_{m-1}(x_i))]}{\partial f_{m-1}(x_i)} \\ &= -\frac{y_i \exp(-y_i f_{m-1}(x_i))}{1 + \exp(-y_i f_{m-1}(x_i))}\end{aligned} \tag{43}$$

所以,

$$\begin{aligned}h_m &= \operatorname*{argmin}_{h\in\mathcal{F}}\sum_{i=1}^{n}[(-\boldsymbol{g}_m)_i - h(x_i)]^2 \\ &= \operatorname*{argmin}_{h\in\mathcal{F}}\sum_{i=1}^{n}\left[\frac{y_i \exp(-y_i f_{m-1}(x_i))}{1 + \exp(-y_i f_{m-1}(x_i))} - h(x_i)\right]^2\end{aligned} \tag{44}$$

2.3 (Optional) What are the similarities and differences between Gradient Boosting and Gradient Descent?

解: 梯度提升算法和梯度下降算法都是在每一轮迭代中, 利用损失函数相对于模型的负梯度方向的信息来对当前模型进行更新, 只不过在梯度下降中, 模型以参数化形式表示, 从而模型的更新是在参数空间进行的. 而在梯度提升中, 模型直接定义在函数空间, 从而模型的更新是在函数空间进行的.

Programming: AdaBoost

The goal of this problem is to give you an overview of the procedure of *AdaBoost*. Here, our "weak learners" are *decision stumps*. Our data consist of $X \in \mathbb{R}^{n \times p}$ matrix with each row a sample and label vector $y \in \{-1, +1\}^n$. A decision stump is defined by:

$$h_{(a,d,j)}(\boldsymbol{x}) = \begin{cases} d, & \text{if } x_j \leqslant a, \\ -d, & \text{otherwise,} \end{cases} \tag{45}$$

where $a \in \mathbb{R}$, $j \in \{1, ..., p\}$, $d \in \{-1, +1\}$. Here $\boldsymbol{x} \in \mathbb{R}^p$ is a vector, and x_j is the j-th coordinate.

Directory of the data is `/code/ada_data.mat`. It contains both a training and testing set of data. Each consists of 1000 samples. There are 25 real valued features for each sample, and a corresponding y label.

3.1 Complete the code skeleton `decision_stump.m` (or `decision_stump()` in `adaboost.py` if you use python). This program takes as input: the data along with a set of weights (i.e., $\{(\boldsymbol{x}_i, y_i, w_i)\}_{i=1}^n$, where $w_i \geqslant 0$ and $\sum_{i=1}^n w_i = 1$), and returns the decision stump which minimizes the weighted training error. Note that this requires selecting both the optimal a, d of the stump, and also the optimal coordinate j.

The output should be a pair $(a^\star, d^\star, j^\star)$ with:

$$l(a^\star, d^\star, j^\star) = \min_{a,d,j} l(a, d, j) = \min_{a,d,j} \sum_{i=1}^n w_i 1_{\{h_{a,d,j}(\boldsymbol{x}_i) \neq y_i\}}. \tag{46}$$

Your approach should run in time $O(pn \log n)$ or better. Include details of your algorithm in the report and analyze its running time.

Hint: you may need to use the function `sort` provided by MATLAB or python in your code, we can assume its running time to be $O(m \log m)$ when considering a list of length m.

3.2 Complete other two code skeletons `update_weights.m` and `adaboost_error.m`. Then run the `adaboost.m`, you will carry out AdaBoost using decision stumps as the "weak learners". (Complete the code in `adaboost.py` if you use python).

3.3 Run your AdaBoost loop for 300 iterations on the data set, then plot the training error and testing error with iteration number as the x-axis.

解: 选择决策树桩时首先对每一维特征进行排序, 复杂度为 $O(n \log n)$, 然后将排序后的数据从小到大对加权的标签积分, 积分值最大的地方就是分界面, 求取分界面的复杂度为 $O(n)$, 特征共 p 维, 因此总的时间复杂度为

$$O(pn \log n + pn) \approx O(pn \log n) \tag{47}$$

课堂上经过理论推导得知, 权重调整系数为

$$\alpha_l = \frac{1}{2} \ln \left(\frac{1 - \epsilon_l}{\epsilon_l} \right) \tag{48}$$

而示例代码中给出 `alpha[i]=np.log((1-e)/e)`, 故将其修改为 `alpha[i]=1/2 * np.log((1-e)/e)`, 否则算法会出现明显的震荡现象.

在数据集上循环 300 次 AdaBoost 算法, 得到训练错误率和测试错误率与循环次数的变化曲线如图 1 所示, 可以看出随着训练集上的错误率逐渐趋于 0, 测试集错误率逐渐趋于 0.1.

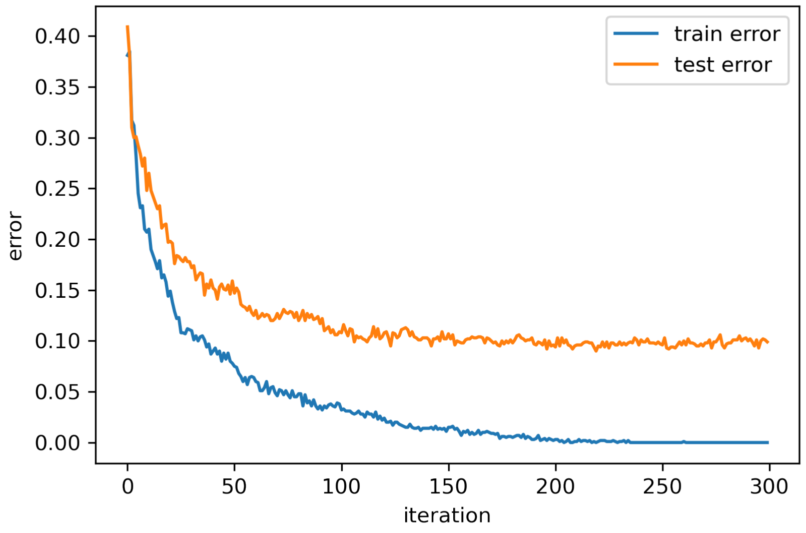

图 1: 训练错误率和测试错误率