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

Ensemble

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

Hong Zhao vzhao@tsinghua.edu.cn

Student:

Problem 1: 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.

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 $m = 1, ..., M$. Then the prediction is given by:

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

Hint: A committee can be viewed as a set of individual models on which we average our predictions.

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

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

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

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

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

The average error made by the models acting individually is therefore:

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

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

$$\begin{aligned} E_{COM} = \quad & \mathbb{E}_{\mathbf{x}}[\{\tfrac{1}{M}\textstyle\sum_{m=1}^{M} y_m(\mathbf{x}) - h(\mathbf{x})\}^2] & (5) \\ = \quad & \mathbb{E}_{\mathbf{x}}[\{\tfrac{1}{M}\textstyle\sum_{m=1}^{M} \epsilon_m(\mathbf{x})\}^2]. & (6) \end{aligned}$$

1.1 Assume that errors have zero mean and are uncorrelated:

$$\begin{aligned} \mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})] = \quad & 0, & (7) \\ \mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})\epsilon_l(\mathbf{x})] = \quad & 0, \qquad m \neq l. & (8) \end{aligned}$$

Please prove that:

$$E_{COM} = \frac{1}{M} E_{AV}. \tag{9}$$

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

$$E_{COM} \leq E_{AV}. \tag{10}$$

1.3 In the previous problem, our error function is $f(y(\mathbf{x}) - h(\mathbf{x})) = (y(\mathbf{x}) - h(\mathbf{x}))^2$(sum-of-square). By making use of *Jensen's inequality*, show that equation (10) holds for any error function $E(y(\mathbf{x}) - h(\mathbf{x}))$ provided it is a convex function of $y(\mathbf{x}) - h(\mathbf{x})$.

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

$$y_{COM}(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}). \tag{11}$$

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

$$y_{min}(\mathbf{x}) \leq y_{COM}(\mathbf{x}) \leq y_{max}(\mathbf{x}). \tag{12}$$

Show that the necessary and sufficient condition for constraint (12) is:

$$\alpha_m \geq 0, \qquad \sum_{m=1}^{M} \alpha_m = 1. \tag{13}$$

Problem 2: 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 $\mathbf{R}$, 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:
   $$(\mathbf{g}_m)_i = \left.\frac{\partial}{\partial f(x_i)} \ell\left(y_i, f(x_i)\right)\right|_{f(x_i)=f_{m-1}(x_i)},$$
   where $\{y_i, x_i\}_1^n$ are n data samples.

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

$$h_m = \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} \left((-\mathbf{g}_m)_i - h(x_i) \right)^2 ,$$

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

$$\nu_m = \arg\min_{\nu > 0} \sum_{i=1}^{n} \ell \left(y_i, f_{m-1}(x_i) + \nu h_m(x_i) \right) ,$$

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

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

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

3. Return f_M.

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

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

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

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

$$h_m = \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} \left[(y_i - f_{m-1}(x_i)) - h(x_i) \right]^2 .$$

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.

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

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

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 $\mathcal{F}$.

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

Problem 3: Adaboost Programming

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:

$$h_{(a,d,j)}(\mathbf{x}) = \begin{cases} d, & if\ x_j \leq a, \\ -d, & otherwise, \end{cases}$$

where $a \in \mathbb{R}$, $j \in \{1, ..., p\}$, $d \in \{-1, +1\}$. Here $\mathbf{x} \in \mathbb{R}^p$ 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 examples. There are 25 real valued features for each example, 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., $\{(\mathbf{x}_i, y_i, w_i)\}_{i=1}^n$, where $w_i \geq 0$ 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:

$$l(a^\star, d^\star, j^\star) = min_{a,d,j} l(a, d, j) = min_{a,d,j} \sum_{i=1}^n w_i 1\{h_{a,d,j}(\mathbf{x}_i) \neq y_i\}. \tag{14}$$

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 $O(mlogm)$ when considering a list of length m.*

3.2 Complete the 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.