EE6550 Machine Learning, Spring 2016

Homework Assignment #3

In this homework assignment, there is no word problems but one programming problem. Please submit your programs, including source code, report and user manual, to iLMS by 13:00 on May 23rd Monday. You are encouraged to consult or collaborate with other students while solving the problems but you will have to turn in your own solutions and programs with your own words. Copying will not be tolerated. If you consult or collaborate, you must indicate all of your consultants or collaborators with their contributions.

The Programming problem: Implementation of the AdaBoost algorithm for binary classification. You should use decision trees of depth one, also known as stumps or boosting stumps as base classifiers for the AdaBoost algorithm. You have to use the MatLab functions in the toolboxes provided by the university version of MatLab to write your code. This means that you have to successfully run your MatLab program in the environment of the university version.

Input:

1. A data file which contains a labeled training sample S. This labeled training sample is used to train AdaBoost algorithm which will return a hypothesis h_S^{AdaBst} after n-fold cross-validation.
2. A data file which contains a labeled testing sample $\tilde{S}$. This labeled testing sample is used to evaluate the performance of the returned hypothesis h_S^{AdaBst}.
3. Choice of n-fold cross-validation, where $n = 5$ or $n = 10$. The free model parameter is the number T of base classifiers needed in the AdaBoost algorithm. As discussed in Lecture 1, we use n-fold cross-validation to determine the best value of the free model parameter T.
   - Randomly partition a given training sample S of m labeled items into n subsamples or folds.
   - $((\omega_{i1}, c(\omega_{i1})), ..., (\omega_{im_i}, c(\omega_{im_i})))$: the ith fold of size m_i, $1 \leq i \leq n$.
     - Usually $m_i = \frac{m}{n}$ for all i.
   - For any $i \in [1, n]$, the AdaBoost algorithm is trained on all but the ith fold to generate a hypothesis h_i, and the performance of h_i is tested on the ith fold.
   - $\hat{R}_{CV}(T)$: the cross-validation error with T base classifiers.

$$\hat{R}_{CV}(T) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{m_i}\sum_{j=1}^{m_i} 1_{h_i(\omega_{ij})\neq c(\omega_{ij})} = \frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{m_i} 1_{h_i(\omega_{ij})\neq c(\omega_{ij})}.$$

   - Choose the value T^* of T which minimizes the cross-validation error $\hat{R}_{CV}(T)$.
   - Train the AdaBoost algorithm with the best number T^* of base classifiers over the full training sample S of size m. The resulted hypothesis will be the returned hypothesis h_S^{AdaBst} from the AdaBoost algorithm.

Output:

1. The optimal value of the number T of base classifiers for $n = 5$ and for $n = 10$ for the n-fold cross-validation.
2. The hypothesis h_S^{AdaBst} returned by the AdaBoost algorithm for $n = 5$ and for $n = 10$ for the n-fold cross-validation.
3. Performance evaluation of the returned hypothesis h_S^{AdaBst} on the labeled testing sample $\tilde{S}$ for $n = 5$ and for $n = 10$ for the n-fold cross-validation.

What to submit? You should submit the following items:

1. The electronic source code of your AdaBoost algorithm. (It is recommended for you to use MatLab to write your programs.)
2. A printed report consisting of at least:
   (a) You should use adult_training.mat and adult_testing.mat given in HW#2 as the training data set and the testing data set respectively.
   (b) A table of the cross-validation error $\hat{R}_{CV}(T)$ as a function of the number T of base classifiers for $n = 5$ and for $n = 10$ for n-fold cross-validation. Discuss how do you determine the optimal value T^* from such a table.
   (c) The hypothesis h_S^{AdaBst} returned by the AdaBoost algorithm with the best number T^* of base classifiers and its performance evaluation on the labeled testing sample $\tilde{S}$ for $n = 5$ and for $n = 10$ for the n-fold cross-validation.
3. A user manual which should include instructions of
   (a) how to compile the source code;
   (b) how to run the algorithm.