THU-70250043, Pattern Recognition (Spring 2018) **Homework: 11**

Ensemble

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

Student: XXX `xxx@mails.tsinghua.edu.cn`

Problem 1

Boosting

Boosting was originally motivated using statistical learning theory, leading to upper bounds on the generalization error. However, these bounds turn out to be too loose to have practical value, and the actual performance of boosting is much better than the bounds alone would suggest. Let's now derive a different and very simple interpretation of boosting in terms of the sequential minimization of an exponential error family.

Consider the exponential error function defined by:

$$E = \sum_{n=1}^{N} exp\{-t_n f_m(\mathbf{x}_n)\} \tag{1}$$

Where $f_m(\mathbf{x})$ is a classifier defined in terms of a linear combination of base classifiers $y_l(\mathbf{x})$ of the form:

$$f_m(\mathbf{x}) = \frac{1}{2}\sum_{l=1}^{m} \alpha_l y_l(\mathbf{x}) \tag{2}$$

and $t_n \in \{-1, 1\}$ are the training set target values. Our goal is to minimize E with respect to both the weighting coefficients α_l and the parameters of the base classifiers $y_l(\mathbf{x})$.

Instead of doing a global error function minimization, however, we shall suppose that the base classifiers $y_1(\mathbf{x}), ..., y_{m-1}(\mathbf{x})$ are fixed, as are their coefficients $\alpha_1, ..., \alpha_{m-1}$, and so we are minimizing only with respect to α_m and $y_m(\mathbf{x})$. Separating off the contribution from base classifier $y_m(\mathbf{x})$, we can then write the error function in the form:

$$\begin{aligned} E &= \sum_{n=1}^{N} exp\{-t_n f_{m-1}(\mathbf{x}_n) - \frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\} && (3) \\ &= \sum_{n=1}^{N} w_n^{(m)} exp\{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\} && (4) \end{aligned}$$

Where the coefficients $w_n^{(m)} = exp\{-t_n f_{m-1}(\mathbf{x}_n)\}$ can be viewed as constants because we are optimizing only α_m and $y_m(\mathbf{x})$. If we denote by Γ_m the set of data points that are correctly classified by $y_m(\mathbf{x})$ and the remaining misclassified points by $\mathcal{M}_m$, then we can in turn rewrite the error function in the form:

$$E = e^{-\alpha_m/2} \sum_{n \in \Gamma_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in \mathcal{M}_m} w_n^{(m)} \tag{5}$$

1.1 From equation (18), show that we are minimizing E with respect to $y_m(\mathbf{x})$ and α_m sequentially while the optimal α_m satisfies $\alpha_m = ln((1-\epsilon_m)/\epsilon_m)$, where:

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}} \tag{6}$$

1.2 Inductively, from equation (17), we can see that, having found α_m and $y_m(\mathbf{x})$, the weights on the data points are updated using:

$$w_n^{(m+1)} = w_n^{(m)} exp\{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\} \tag{7}$$

Show that the weights $w_n^{(m)}$ are updated at the next iteration as:

$$w_n^{(m+1)} = w_n^{(m)} exp\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\} \tag{8}$$

Hint: finally, once all the base classifiers are trained, new data points are classified by evaluating the sign of the combined function according to equation (15). Since the factor $1/2$ *doesn't affect the sign, it can be omitted. The whole procedure above gives us a brief insight of adaboost algorithm.*

Problem 2

Adaboost Programming

The goal of this problem is to give you an overview of the procedure of *Adaboost*. In problem 4, you will implement more sophisticated classifiers under the sketch of *Adaboost*, but the basic ideas are completely the same.

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.

2.1 Complete the code skeleton **decision_stump.m**. 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 $w_i \geq 1$ 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:

$$l(a^\star, d^\star, j^\star) = min_{a,d,j} l(a, d, j) = min_{a,d,j} \sum_{i=1}^n w_i 1\{h_{a,d,j}(\mathbf{x}_i) \neq y_i\} \tag{9}$$

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 in your code, we can assume its running time to be $O(mlogm)$ when considering a list of length m.*

2.2 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”.

2.3 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.