Problem 1 – Boosting algorithms on stumps

Read the whole problem carefully before you start working. You need to hand out the required plots (described at the end) and your comments, including the answers to the questions scattered through the text.

Code: For the implementations in questions 1,2,3,5 you are required to write your own code.

This homework will make use of the following (two-dimensional) data sets: hw2-linear-train.dat, hw2-circle-train.dat, hw2-linear-test.dat, hw2-circle-test.dat having each 200 examples. The first two are training sets, which you will use to construct your classifiers. The last two are test sets, on which you will evaluate the performance of the classifiers you obtained.

1 - Fitting a “stump” Write code that implements the following classifier.

Input $d$-dimensional data $x^1, \ldots, x^N$

It is assumed that $\{x^1_j, \ldots, x^N_j\} \subseteq [A_j, B_j]$ with $R_j = B_j - A_j$. In other words, the range of coordinate $j$ has value $R_j$, given. (For both data sets in this assignment, all the ranges $R_j$ are 1.)

the index of a coordinate $j \in \{1, 2, \ldots, d\}$

weights $w_i$, $i = 1, \ldots, N$ for the datapoints

(they are non-negative and normalized)

1. Choose a coordinate $j = 1 : d$

2. Sort the data in increasing order by coordinate $x^i_j$; denote by $x^{(i)}_{\{\}}$ the order statistics; reorder the labels $\{y_i\}$ and the weights $\{w_i\}$ accordingly

3. for $i = 0 : N$

   Calculate $\epsilon_i = -\sum_{i' \leq i} y^{(i')} w^{(i')} + \sum_{i' > i} y^{(i')} w^{(i')}$. This represents the fraction of correctly classified points - fraction of errors for the stump $f(x) = x_j - b_i$ with $b_i \in (x^{(i)}_j, x^{(i+1)}_j)$ along direction $j$. 


4. Let $E_+ = \max_i \epsilon_i$, $l_+ = \arg\max_i \epsilon_i$, $E_- = \max_i (-\epsilon_i)$, $l_- = \arg\max_i (-\epsilon_i)$.

\[ f(x) = \begin{cases} 
\frac{x_j - \frac{x_j^{(l_+)} + x_j^{(l_+ + 1)}}{2}}{R_j} & \text{if } E_+ > E_- \\
-\frac{x_j - \frac{x_j^{(l_-)} + x_j^{(l_- + 1)}}{2}}{R_j} & \text{if } E_+ \leq E_- 
\end{cases} \]

In other words, this classifier is a hyperplane perpendicular on coordinate axis $j$ passing halfway between two data points. The normalization by $R_j$ ensures that the values of $f$ are bounded in $[-1, 1]$ for all input data.

Let $\mathcal{F}_j$ be the family of stumps for a fixed $j$ and let $\tilde{\mathcal{F}}_j = \{ \text{sign} f, f \in \mathcal{F}_j \}$ be the corresponding family of integer-valued stumps.

Note that for some data sets, the same value of $E_\pm$ can be attained for several different stumps along the same direction. In that case, choose one of them arbitrarily.

2. Run the Discrete AdaBoost algorithm on the hw2-linear-train.dat, hw2-circle-train.dat for $M = 200$ iterations, using the families $\tilde{\mathcal{F}}_1, \tilde{\mathcal{F}}_2$ as base classifiers. For simplicity, take $j = 1$ at iterations $k = 1, 3, 5, \ldots$ and $j = 2$ on the even iterations.

3. Run the following Real AdaBoost algorithm on hw2-linear-train.dat, with base classifiers from $\mathcal{F}_{1,2}$. As before, alternate the direction of the stumps (take $j = 1$ for $k = 1, 3, 5, \ldots$ and $j = 2$ otherwise) and run for $M = 200$ boosting iterations.

\textbf{Initialize} \quad F = 0, \text{ weights } w_i = \frac{1}{N} \\
\text{for} \quad k = 1, 2, \ldots, M \\
\quad \text{estimate the real valued classifier (stump) } f_k \in \mathcal{F}_{j_k} \text{ from the weighted data} \\
\quad \text{estimate the error } r_k = \sum_i w_i y_i f_k(x^i), \quad e_k = \frac{1}{2} r_k \quad \text{set } c_k = \frac{1}{2} \log \frac{1-e_k}{e_k} \\
\quad \text{set } w_i \leftarrow w_i \exp[-c_k y_i f_k(x^i)] \text{ and renormalize} \\
\textbf{Output} \quad F(x) = \sum_{k=1}^{M} c_k f_k(x)

(This is the algorithm of Schapire & Singer, “Improved boosting algorithms for confidence rated predictions”, Machine Learning, 37(3), 1999.)

4. What would happen if RealAdaBoost was run on the hw2-circle-train.dat? More precisely, would the training error of $E_k$ decrease to 0? Explain why or why not. [OK to find the answer by experiment, i.e. by actually running RealAdaBoost on the the circle data, but not required.]

5. Run Real AdaBoost again on hw2-linear-train.dat, now with pruning of the small weights. At each step, if $w_i < 1/10N$, set $w_i = 0$, then renormalize
6. Plots and comments:

- Plot the two training sets in 2D.

- Plot $\epsilon_i$ vs $i$ for $k = 1$ or $2$, for each of the two data sets (2 plots, one for each data set).

- For each of the 3 algorithms and for each of the classification problems you have trained them on, let $F_k = \sum_{m=1}^{k} c_m f_m$ (the current boosted classifier). Let $E_k = \frac{1}{N} \sum_{i=1}^{N} 1_{F_k(x^i) \neq y^i}$ the training error of $F_k$. Define $E_{test}^k$ similarly, as the average number of errors of $F_k$ on the test set. Plot the $E_k$ and $E_{test}^k$ vs $k$; if possible, on the same plot for the 3 algs. Comment on the plots: Which of the 3 algorithms performs best? Do you see any evidence of overfitting?

- For Real AdaBoost only, plot the cumulative distribution (or histogram) of the margins $y^i F(x^i)$ at $k = 10, 50, 200$. Make these on the same plot, or align the plots for easy comparison. (I recommend the cumsum over the histogram because it makes more legible plots.) Comment on what changes you observe as $k$ grows.

- For Real AdaBoost with weight pruning, plot the number of non-zero weighted points vs $k$.

- For the best of the three classifiers $F_M$ on the linear data set, and for Discrete AdaBoost on the circle data set, plot the decision regions or decision boundary.