		Hypothesis 1 (1st iteration)				Hypothesis 2 (2nd iteration)			
i	Label	D_0	$f_1 \equiv [x > 2]$	$f_2 \equiv [y > 6]$	$h_1 \equiv [x > 2\ ]$	D_1	$f_1 \equiv [x > 11]$	$f_2 \equiv [y > 11]$	$h_2 \equiv [y > 11]$
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
1	−	$\frac{1}{10}$	-	+	-	$\frac{1}{16}$	-	-	-
2	−	$\frac{1}{10}$	-	-	-	$\frac{1}{16}$	-	-	-
3	+	$\frac{1}{10}$	+	+	+	$\frac{1}{16}$	-	-	-
4	−	$\frac{1}{10}$	-	-	-	$\frac{1}{16}$	-	-	-
5	−	$\frac{1}{10}$	-	+	-	$\frac{1}{16}$	-	+	+
6	+	$\frac{1}{10}$	+	+	+	$\frac{1}{4}$	-	-	-
7	+	$\frac{1}{10}$	+	+	+	$\frac{1}{16}$	+	-	-
8	−	$\frac{1}{10}$	-	-	-	$\frac{1}{16}$	-	-	-
9	+	$\frac{1}{10}$	-	+	-	$\frac{1}{4}$	-	+	+
10	+	$\frac{1}{10}$	+	+	+	$\frac{1}{16}$	-	-	-

Table 1: Table for Boosting results

CM146 HW#4

Zhaoxing Deng, 005024802

1 Boosting

Consider the following examples $(x, y) \in \mathbb{R}^2$ (i is the example index):

i	x	y	Label
1	0	8	−
2	1	4	−
3	3	7	+
4	-2	1	−
5	-1	13	−
6	9	11	−
7	12	7	+
8	-7	-1	−
9	-3	12	+
10	5	9	+

(a) See Table 1.

(b) See Table 1.

(c) See Table 1.

(d) In the first iteration, the error $\epsilon_1 = \frac{2}{10}$. Then $\alpha_1 = \frac{1}{2}\log(\frac{1-\epsilon_1}{\epsilon_1}) = 1$. So we can get $D_1(i) = \frac{D_0(i)}{Z_1}2^{-\alpha_1 y_i h_1(x_i)} = \begin{cases} \frac{1}{4} & incorrect \\ \frac{1}{16} & correct \end{cases}$.

In the next iteration, the error $\epsilon_2 = \frac{4}{16}$. Then $\alpha_2 = \frac{1}{2}\log(\frac{1-\epsilon_2}{\epsilon_2}) = \frac{\log 3}{2}$. The the final hypothesis produced by AdaBoost is $H_{final} = sgn(\sum \alpha_t h_t(x)) = \begin{cases} 1 & x > 2 \\ -1 & x \leq 2 \end{cases}$

2 Multi-class classification

(a) i. One vs. All learns k classifiers. All vs. All learns C_2^kclassifiers.

ii. One vs. All uses m examples for each classifier,. All vs. All uses $\frac{2m}{k}$examples for each classifier.

iii. For One vs. All, we will decide using the label of the classifier which has the highest value for that point. $\hat{y} = argmax_{y \subseteq Y} f(\mathbf{y}; \mathbf{w}, \mathbf{x})$. For All vs. All, we will decide using the label if that lable has more votes than others. We can also use A tournament, which starts with n/2 pairs and only continues with winners.

iv. The complexity of One vs. All is $O(k)$, because it needs to train k classifiers. The complexity of All vs. All is $O(k^2)$, since it needs to train scales withC_2^k classifers which has the complexity of k^2.

(b) I would prefer One vs. All. It's easy to implement and has a good performance in practice. All vs. All will need large space to store model. It has much more classifers to train. It may also overfit the trainning set.

(c) Kernel Perceptron may have the ability to separate data which is not linearly separable in the original space, so every subset of trainning data is seperable. Because Kernel perceptron needs to compute the kernel between every pair of points, it has complexity of $O(n^2)$ on n examples. Then One vs. All has $O(k * n * n) = O(kn^2)$ complexity and All vs. All has$O(k * (k-1)/2 * n/k * n/k) = O(n^2)$ complexity. So All vs. All will have less time for the training. Thus I would perfer All vs. All.

(d) The overall training time complexity of One vs. All is$O(kdn^2)$, and All vs. All is $O(k * (k-1)/2 * n/k * n/k * d) = O(dn^2)$. Thus the All vs. All is more efficient.

(e) One vs. All is $O(kd^2n)$, and All vs. All is $O(k * (k-1)/2 * n/k * d^2) = O(kd^2n)$. Thus they are equally efficient.

(f) For Counting,$O(d * m * (m-1)/2) = O(dm^2)$. For Knockout, $O(d * (m-1)) = O(dm)$. Thus knockout is more efficient.