CM146, Winter 2018
Problem Set xx: yy

1 Problem 1

(a) Problem 1a **Solution:**

		Hypothesis 1 (1st iteration)				Hypothesis 2 (2nd iteration)			
i	Label	D_0	$f_1 \equiv [x > 2\ ]$	$f_2 \equiv [y > 6\ ]$	$h_1 \equiv$ [sgn(x - 2)]	D_1	$f_1 \equiv [x > 10\ ]$	$f_2 \equiv [y > 11\ ]$	$h_2 \equiv$ [sgn(y - 11)]
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
1	−	0.1	-	+	-	0.0439	-	-	-
2	−	0.1	-	-	-	0.0439	-	-	-
3	+	0.1	+	+	+	0.0439	-	-	-
4	−	0.1	-	-	-	0.0439	-	-	-
5	−	0.1	-	+	-	0.0439	-	+	+
6	−	0.1	+	+	+	0.3244	-	-	-
7	+	0.1	+	+	+	0.0439	+	-	-
8	−	0.1	-	-	-	0.0439	-	-	-
9	+	0.1	-	+	-	0.3244	-	+	+
10	+	0.1	+	+	+	0.0439	-	-	+

For the D_0, we use a uniform distribution. $\Rightarrow D_0 = \frac{1}{10} = 0.1$

(b) The error for each of the weak learners f_1, f_2 are as follows:

$$\epsilon_1 = \sum_{incorrect} D_0 = (0.1)(2) = 0.2$$

$$\epsilon_2 = \sum_{incorrect} D_0 = (0.1)(3) = 0.3$$

Hence, the algorithm chooses f_1 as its hypothesis in the first step

$$\alpha_1 = \frac{1}{2}\log_2\left(\frac{1-0.2}{0.2}\right) = \frac{1}{2}\log_2(4) = 1$$

(c) To Calculate the new weights for each example, we can divide it into examples we predicted correctly and the ones we predicted incorrectly

$$D_{1,correct} = \frac{0.1}{Z_1} \cdot 2^{-1} = \frac{0.05}{Z_t}$$

$$D_{1,incorrect} = \frac{0.1}{Z_1} \cdot 2^{1} = \frac{0.2}{Z_t}$$

$$Z_t = 8(0.05) + 2(0.4) = 0.8$$

$$D_{1,correct} = \frac{0.05}{0.8} = 0.0625$$

$$D_{1,incorrect} = \frac{0.2}{0.8} = 0.25$$

$$\epsilon_1 = 0.0625(2) + 0.25 = 0.375$$

$$\epsilon_2 = 0.0625(4) = 0.25$$

$$\alpha_2 = \frac{1}{2}\log_2\left(\frac{1-0.25}{0.25}\right) = 0.7924$$

(d) The final hypothesis learned by AdaBoost after 2 iterations is:

$$h = sign(sign(x-2) + 0.7924 \cdot sign(y-11))$$

2 Problem 2

Solution: Solution to problem 2

(a) i. For One-vs All: Number of Classifiers = K
For All-vs-All: Number of Classifiers = $\frac{K(K-1)}{2}$

ii. For One-vs-All: Need to look at all m examples during training time
For All-vs-All: Need to only look at $\frac{2m}{k}$ during training time for each classifier, ($\frac{m}{k}$ for each of the classes to which the classifier corresponds to)

iii. For One-vs-all: Since the perceptron algorithm learns a real valued function, to predict the class for each example, we just pick the class that has the highest value from the perceptron algorithm.
For all-vs-all: For all-vs-all classification, we have the option of Majority vote or Tournament style. For Majority, for each example, pick the class that the maximum number of classifiers picked (i.e. the majority of the classifier, although it can be less than 50%). For Tournament style, we divide the classifiers into pairs, and the winner of each pair moves on and we keep repeating that until we have a winner.

iv. Complexity for One-vs-all: O(KL)
Complexity for All-vs-All: O(K^2L)
where L is the time-complexity of the learning algorithm we're using on number of training examples used (answer to part ii)

(b) The preference for either scheme depends on the classification problem and the efficiency of the training algorithm. For instance, if the data is not linearly separable, we would prefer to use one vs all, instead of all vs all. However, since the number of examples each classifier trains on is lesser with All vs All, will make the training time per classifier faster, even though there will be more classifiers that we would have to train. Overall, the preference depends on the classification problem, the data and the hardware we have available to us for this problem.

(c) Yes, using a Kernel Perceptron Algorithm changes the analysis. The Kernel Perceptron algorithm runs in time O(n^2d), where n is the number of examples provided, and d is the dimension of the kernel function. Using this knowledge,

Runtime for one-vs-all: O(Km^2d), where d is the dimension of the feature vectors
Runtime for All-vs-All: O(K^2) * O(d $\left(\frac{2m}{K}\right)^2$) = O($m^2d$) where d is the dimension of the feature vectors
Thus, All-vs-All is more efficient, but one-vs-all might be preferred depending on the learning problem.

(d) This learning algorithm has the same time complexity as the Kernel Perceptron, and hence the runtime is the same as part (c).
Runtime for one-vs-all: O(Km^2d)
Runtime for All-vs-All: O(m^2d)
Thus, All-vs-All is more efficient, but one-vs-all might be preferred depending on the learning problem.

(e) Runtime for one-vs-all: O(K) classifiers, that will train in O(d^2m) time. $\Rightarrow O(Kd^2m)$
Runtime for All vs all: O(K^2) classifiers that will train in O($d^2\frac{m}{k}$) time. $\Rightarrow O(Kd^2m)$
Hence, both the schemes have the same run time, and are equally efficient.

(f) Runtime for counting: For counting, we need to go through all O(m^2) classifiers, each of which has a runtime of O(d). $\Rightarrow O(m^2d)$
Runtime for Knockout: For knockout, we compare two classes at a time using the classifier that corresponds to these two classes. Once a class "loses", we don't consider it again. Hence we only need to go through O(m) classifiers, each of which has a runtime of O(d). $\Rightarrow O(md)$