Problem 1  Thrilled by the analysis of the AdaBoost algorithm, you ran home from class to code up the algorithm. But the phone rings when you’re typing step 2(b), and accidentally your implementation just sets $\alpha_t = 1$. This of course affects the update rule defining $D_{t+1}$ (since that involves $\alpha_t$), and similarly in the final line of your implementation, since $\alpha_t = 1$ for each $t$, you have $f(x) = \sum_{t=1}^{T} h_t(x)$; so your final hypothesis $H$ just takes an unweighted majority vote over $h_1, \ldots, h_T$.

Fortunately, for you, you also have access to a pretty great weak learner, which always, when run on any distribution $D$ over your examples, generates a weak hypothesis which has advantage at least $1/4$.

You run AdaBoost on a dataset of $m = 10^6$ examples, and with a slight (intentional) tweak in the algorithm: instead of having the algorithm loop from $t = 1, \ldots, T$, you structure the loop as “if $H$ misclassifies at least one example, do another iteration of the loop.” (In other words, you’re trying to use AdaBoost as a consistent hypothesis finder.)

What happens? Might your algorithm run forever, or can you guarantee that it will halt after some number of iterations of the loop? If you think it may run forever, explain why. If you think it will halt after some number of iterations of the loop, give the best bound you can on the maximum # of iterations of the loop that may be executed. (You can use a calculator/computer/etc to come up with your bound.)

Problem 2  Let $C$ be any concept class over $\{0, 1\}^n$. Show that if $C$ is efficiently PAC learnable, then there is an efficient algorithm that, given $0 < \delta \leq 1$ and a sample $S$ of $m$ examples labeled according to some concept $c$ in $C$, outputs with probability at least $1 - \delta$ a hypothesis $h$ such that

(i) $h$ is consistent with $S$, and
(ii) $\text{size}(h) \leq p(n, \text{size}(c), \log m, 1/\delta)$ for some polynomial $p$.

(Hint: Use the AdaBoost algorithm together with the efficient PAC learning algorithm for $C$. You may define the size of a hypothesis $h$ as you wish for this problem, provided that your definition is reasonable, and you may assume, for the purposes of this problem, that each real number which occurs in the representation of $h$ contributes 1 to its size.)

(A cultural note unrelated to solving the problem: recall that the Occam’s Razor theorem can be viewed as showing that “compression implies learnability.” This problem essentially shows a converse in a very strong sense – note that the hypothesis $h$ which encodes all $m$ of the correct labels for the examples in $S$ is of size only logarithmic in $m$ (ignoring other parameters).)

**Problem 3** Let \( C \) be any concept class. Suppose that algorithm \( A \) is an efficient proper PAC learning algorithm for \( C \) in the noise-free setting (i.e. given access to an example oracle \( EX(c,D) \), algorithm \( A \) outputs a hypothesis \( h \) which belongs to \( C \) and satisfies the PAC criteria). Suppose that moreover there is an efficient PAC learning algorithm \( B \) for concept class \( C \) in the presence of random classification noise, but \( B \) is not a proper PAC learning algorithm (i.e. the hypotheses which \( B \) outputs belong not to \( C \), but to some other hypothesis class \( H \)). Show that then there must exist an efficient proper PAC learning algorithm for \( C \) in the presence of random classification noise.

**Problem 4** Recall that in the malicious noise model with noise rate \( \eta \), the example oracle flips a biased coin and with probability \( \eta \) returns an arbitrary example in the domain labeled arbitrarily. With probability \( 1 - \eta \) it draws a random example according to \( D \) and passes it with the correct label to the learner.

Suppose that concept class \( C \) is learnable to accuracy \( \epsilon \) and confidence \( \delta \) in the PAC model by a polynomial time algorithm which has sample size \( m(\epsilon, \delta) \). Let \( s = m(\epsilon/8, 1/2) \). Show that there is a polynomial time algorithm which PAC learns \( C \) in the presence of malicious noise at rate \( \eta \), where \( \eta = \min\{\epsilon/8, \ln s / s\} \).

**Problem 5** Our definition of efficient PAC learning in the presence of random classification noise at rate \( \eta < 1/2 \) requires that the algorithm run in time \( \text{poly}(\frac{1}{1-2\eta}) \) (ignore all the other parameters for simplicity for this problem). This is intuitively plausible, since (i) if \( \eta = 0 \) (no noise) the function \( \frac{1}{1-2\eta} \) equals 1, and (ii) as \( \eta \) approaches \( 1/2 \) (and learning becomes impossible) the function \( \frac{1}{1-2\eta} \) approaches infinity. But why is \( \frac{1}{1-2\eta} \) the “right” function, as opposed to some other function such as \( 1 + \log(\frac{1}{1-2\eta}) \) or \( \exp(\frac{1}{1-2\eta} - 1) \), that satisfies (i) and (ii)?

Argue as clearly and convincingly as you can that any PAC learning algorithm for learning in the presence of random classification noise at rate \( \eta \) must have runtime which grows as \( \Omega(\frac{1}{1-2\eta}) \). (Hint: One way to show this is to show that the sample complexity must grow as \( \Omega(\frac{1}{1-2\eta}) \).)