# Homework 1

Homework rules:

- Homeworks are due one week after they are handed out, unless otherwise specified. *No late homeworks accepted!*
- Feel free to talk to other members of the class in doing your homework. I'm more concerned that you learn how to solve the problem, than that you demonstrate that you solved it entirely on your own. You should, however, write up your solution yourself. Please try to keep solutions brief and clear. You may refer to the literature, but do not refer to handouts from previous years' classes (in case I repeat some problems from previous years).

**Due Wednesday, September 21**

## Problem 1.

Suppose that $f_1, f_2, \ldots$ is an enumeration of some infinite set of computable total 0-1-valued functions. More precisely, this enumeration is computed by some program $M$; the output of $M$ on input $i$ is a program $P_i$ that computes $f_i$.

Suppose you are learning an unknown function $f$, which is guaranteed to be one of the $f_i$'s. You are shown a sequence of input/output pairs that is consistent with the unknown function. For each pair, you are shown the input and then, before you see the actual output, you are asked to guess the output. Then you are shown the output. If you guessed wrong, you are said to have made a prediction mistake.

Show how to learn $f$ in such a way that you make only $O(\lg t)$ prediction mistakes, where $t$ is the least index such that $f = f_t$.

## Problem 2.

Consider the following class $C_{cmaj}$ of functions. (Let's call them *crypto-majority* functions for convenience.) The instance space is $X = \{0,1\}^n$. Let $a \oplus a'$ denote the "exclusive-or" of bits $a$ and $a'$. Each crypto-majority function $c_b$ is defined by a vector $b = (b_1, b_2, \ldots, b_n) \in \{0,1\}^n$ as follows: For each instance $x = (x_1, x_2, \ldots, x_n)$, let $y = (y_1, y_2, \ldots, y_n)$, where each $y_i = x_i \oplus b_i$. Then $c_b(x)$ is 1 if and only if at least half of the bits $y_i$ are 1.

(a) Show that there is an efficient (polynomial-time) on-line learning algorithm for learning $C_{cmaj}$ that makes a number of prediction mistakes that is at most polynomial in $n$. The learner knows *a priori* that the target function is a crypto-majority function, but doesn't know the vector $b$.

(b) Show that $C_{maj}$ can also be learned efficiently in a "membership query model". Here the learner is allowed to ask, for vectors $x$ of his choice, whether or not $x$ is a positive example of the target concept $c_b$. Show that a learner can exactly identify $c_b$ by asking only a polynomial (in $n$) number of such membership queries.