Homework 1 Solutions

Each problem was worth 10 points. The average score was out of 13.6 out of 20.

Problem 1.

Since the enumeration is unbounded, we cannot apply the halving algorithm directly. Instead, we will apply the halving algorithm to successively larger blocks of the enumeration until the function f_t is found. A simple, though somewhat wasteful, solution is as follows. In each phase $i \geq 0$, consider the block of functions $B_i = \{f_j : 1 \leq j \leq 2^{2^i}\}$. Apply the halving algorithm to B_i. If all functions in B_i are eventually eliminated, then repeat the process on the next largest block. Since $B_i \subset B_{i+1}$, there will clearly be some repetition of work. However, the analysis will show that this repetition does not hurt us.

This process will terminate when the first block B_k is reached such that $f_t \in B_k$. Thus, $k = \lceil \lg \lg t \rceil$. The total number of prediction mistakes made will be the sum of the prediction mistakes made on each of the blocks $B_0, B_1, B_2, \ldots, B_k$. For every $i < k$, the number of prediction mistakes made on block B_i will be at most $\lg |B_i| + 1 = 2^i + 1$. The number of prediction mistakes made on block B_k will be at most $\lg |B_k| = 2^k$. Thus, the maximum total number of prediction mistakes is given by

$$\begin{aligned}
(2^0 + 1) + (2^1 + 1) + \cdots + (2^{k-1} + 1) + 2^k &= 2^{k+1} + k - 1 \\
&= 2^{\lceil \lg \lg t \rceil + 1} + \lceil \lg \lg t \rceil - 1 \\
&< 2^{\lg \lg t + 2} + \lg \lg t \\
&= 4 \lg t + \lg \lg t \\
&= O(\lg t)
\end{aligned}$$

Problem 2.

(a) We will learn the class of crypto-majority functions using Littlestone's Winnow2 algorithm. The Winnow algorithm works for Boolean functions which are linearly separable. That is, for instance space $X = \{0,1\}^N$, it works for a concept class C, for which there exists a δ, where $0 < \delta \leq 1$, such that the following condition holds: For each $c \in C$, there exists $w = (w_1, w_2, \ldots, w_N)$, where each $w_i \geq 0$, such that for all $x = (x_1, x_2, \ldots x_N) \in X$,

$$w \cdot x = \sum_{i=1}^{N} w_i x_i \geq 1, \text{ if } c(x) = 1 \tag{1}$$

$$w \cdot x = \sum_{i=1}^{N} w_i x_i \leq 1 - \delta, \text{ if } c(x) = 0 \tag{2}$$

Recall that 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 $x = (x_1, x_2, \ldots, x_n)$, we define $y = (y_1, y_2, \ldots, y_n)$, 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. In other words, $c_b(x) = 1$ if and only if $\sum_{i=1}^{n} y_i \geq n/2$. Note that by the definition of exclusive-or, each $y_i = \overline{x}_i b_i \vee x_i \overline{b}_i$. Moreover, since terms $\overline{x}_i b_i$ and $x_i \overline{b}_i$ cannot be true at the same time, each $y_i = \overline{x}_i b_i + x_i \overline{b}_i$.

We now transform the problem as follows. Let $N = 2n$. That is, we will double the number of attributes, just as we doubled the number of attributes when learning arbitrary disjunctions using Winnow1. For each example $x = (x_1, x_2, \ldots, x_n)$, we let

$$x' = (x'_1, x'_2, \ldots, x'_n, x'_{n+1}, \ldots, x'_N) = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n, x_1, x_2, \ldots, x_n)$$

Now let

$$b' = (b'_1, b'_2, \ldots, b'_n, b'_{n+1}, \ldots, b'_N) = (b_1, b_2, \ldots, b_n, \overline{b}_1, \overline{b}_2, \ldots, \overline{b}_n)$$

Then,

$$\begin{aligned} x' \cdot b' &= \sum_{i=1}^{N} x'_i b'_i \\ &= \sum_{i=1}^{n} \overline{x}_i b_i + \sum_{i=1}^{n} x_i \overline{b}_i \\ &= \sum_{i=1}^{n} (\overline{x}_i b_i + x_i \overline{b}_i) \\ &= \sum_{i=1}^{n} y_i \end{aligned}$$

Thus, $c_b(x) = 1$ whenever $x' \cdot b' \geq n/2$, or equivalently whenever $2/n(x' \cdot b') \geq 1$. Moreover, $c_b(x) = 0$ whenever $2/n(x' \cdot b') \leq 1 - 2/n$. Thus, letting $\delta = 2/n$ and $w = (w_1, w_2, \ldots, w_N)$, where each $w_i = 2b'_i/n$, we satisfy equations 1 and 2. Thus, the crypto-majority functions are linearly separable with a minimum separation of $\delta = 2/n$.

We can now apply the Winnow2 learning algorithm to the transformed problem. The total number of mistakes is bounded by

$$\frac{8}{\delta^2}\frac{N}{\theta} + \left(\frac{5}{\delta} + \frac{14 \ln \theta}{\delta^2}\right) \sum_{i=1}^{N} w_i$$

Note that $\sum_{i=1}^{N} w_i \leq 2$. Plugging in, we see that the total number of mistakes is at most $O(n^3)$.

This class of functions can also be learned using the perceptron convergence algorithm.

(b) We show how to learn crypto-majority functions with at most $O(n)$ queries. We will do this in two steps:

1. Find an example x such that $\sum(x_i \oplus b_i)$ is exactly $\lceil n/2 \rceil$.
2. Learn each bit of b_i using examples which differ from this example in only one bit.

Let's assume without loss of generality that n is even. Then in the first step, we are trying to find an example which differs from b in exactly half of its bits. We do this as follows. First we ask whether an arbitrary example, say $y = (1, 1, 1, \ldots, 1)$, is a positive or negative example. Suppose it is a positive example. Then this example differs from b in at least $n/2$ bits. Now we ask whether the complement of this example (in this case, $\overline{y} = (0, 0, \ldots, 0)$) is a positive or negative example. Note that if y differs from b in i bits than $\overline{y}$ differs from b in $n - i$ bits. Thus, if $\overline{y}$ is also a positive example, then both y and $\overline{y}$ differ from b in exactly $n/2$ bits, and we are done with step 1. Otherwise, we will transform example y to example $\overline{y}$ bit by bit, and ask membership queries each time. That is, we will query the following examples in order: $(0, 1, 1, \ldots, 1), (0, 0, 1, 1, \ldots, 1), (0, 0, 0, 1, \ldots, 1)$, etc., stopping when we reach the first negative example. Note that since this is a negative example, fewer than half of its bits differ from b. On the other hand, the example previously queried differs from it in only one bit and is a positive example. Thus that previous example has exactly $n/2$ bits differing from b, and we have completed the first step of the algorithm. This process takes at most $n + 1$ queries.

Now that we have found an example x which differs from b in exactly $n/2$ bits, we proceed as follows. Any example which differs from example x in exactly one bit has either exactly $n/2 + 1$ bits different from b, or exactly $n/2 - 1$ bits different from b. Let $x^{(i)}$ denote an example which is the same as example x in all bits except bit i. In the i-th bit, $x^{(i)}$ has the complement bit. Then, if $x^{(i)}$ is a positive example, we know that it has $n/2 + 1$ bits different from b, and in particular we know that the i-th bit is different from b_i. Thus we know that $b_i = x_i$. On the other hand, if $x^{(i)}$ is a negative example, we know that it has $n/2 - 1$ bits different from b, and in particular we know that the i-th bit is the same as b_i. Thus we know that $b_i = \overline{x}_i$. Thus asking n queries for all $x^{(i)}$ determines all bits of b.

We ask a total of at most $2n + 1$ queries.