Problem 1

[10] As on Problem 1 on Homework #1, let $X = \mathbb{R}$, and let $C_s$ be the class of concepts defined by unions of $s$ intervals. Compute the VC-dimension of $C_s$ exactly.

Problem 2

[15] For $i = 1, \ldots, n$, let $G_i$ be a space of concepts ($\{0, 1\}$-valued functions) defined on some domain $X$, and let $F$ be a space of concepts defined on $\{0, 1\}^n$. (That is, each $g_i \in G_i$ maps $X$ to $\{0, 1\}$, and each $f \in F$ maps $\{0, 1\}^n$ to $\{0, 1\}$.) Let $H$ be the space of all concepts $h : X \to \{0, 1\}$ of the form

$$h(x) = f(g_1(x), \ldots, g_n(x))$$

for some $f \in F$, $g_1 \in G_1, \ldots, g_n \in G_n$.

Give a careful argument proving that

$$\Pi_H(m) \leq \Pi_F(m) \cdot \prod_{i=1}^n \Pi_{G_i}(m).$$

[An optional continuation of this problem, applicable to feedforward networks, is given in Problem 5.]

Problem 3

[15] Show that Sauer’s Lemma is tight. That is, for each $d = 0, 1, 2, \ldots$, give an example of a class $C$ with VC-dimension equal to $d$ such that for each $m$,

$$\Pi_C(m) = \sum_{i=0}^d \binom{m}{i}.$$

Problem 4

This problem explores another general method for bounding the error when the hypothesis space is infinite.

Some algorithms output hypotheses that can be represented by a small number of examples from the training set. For instance, suppose the domain is $\mathbb{R}$ and we are learning a (positive) half-line of the form $x \geq a$ where $a$ is a threshold defining the half-line. A simple algorithm chooses the leftmost positive training example and outputs the half-line defined by using this point as a threshold, which is clearly consistent with the training data. Thus, in this case, the hypothesis can be represented by just one of the training examples.

More formally, let $F$ be a function mapping labeled examples to concepts, and assume that algorithm $A$, when given training examples $(x_1, c(x_1)), \ldots, (x_m, c(x_m))$ labeled by some unknown $c \in C$, chooses some $i_1, \ldots, i_k \in \{1, \ldots, m\}$ and outputs a hypothesis $h = F((x_{i_1}, c(x_{i_1})), \ldots, (x_{i_k}, c(x_{i_k})))$ which is consistent with all $m$ training examples. In a sense, the algorithm has “compressed” the sample down to a sequence of just $k$ of the $m$ training examples. (We assume throughout that $m > k$.) For instance, in the example of half-lines, $k = 1$; $i_1$ is the index of the leftmost positive training example; and the function $F$ returns a half-line hypothesis with threshold $x_{i_1}$, that is, the hypothesis $h = F((x_{i_1}, c(x_{i_1})))$ which classifies $x$ positive if and only if $x \geq x_{i_1}$. 
a. [5] Give such an algorithm for axis-aligned hyper-rectangles in $\mathbb{R}^n \text{ with } k = O(n)$. (An axis-aligned hyper-rectangle is a set of the form $[a_1, b_1] \times \cdots \times [a_n, b_n]$, and the corresponding concept, as usual, is the binary function that is 1 for points inside the rectangle and 0 otherwise. For $n = 2$, this is the class of rectangles used repeatedly as an example in class.) Your algorithm should run in time polynomial in $m$ and $n$.

b. [15] Returning to the general case, assume as usual that the examples are chosen at random from some distribution $D$. Also assume that the size $k$ is fixed. Argue carefully that the error of the output (and consistent) hypothesis $h$, with probability at least $1 - \delta$, satisfies the bound:

$$\text{err}_D(h) \leq O\left(\frac{\ln(1/\delta) + k \ln m}{m - k}\right).$$

[Side note: A difficult, long-standing open problem asks if it is always possible to find such a “compression scheme” whose size $k$ is equal to (or proportional to) the VC-dimension $d$ of the target class $C$.]

Problem 5 – Optional (Extra Credit)

[15] This problem shows one way in which the methods we have been developing can be applied to feedforward networks, including (some) neural networks.

A feedforward network, as in the example above, is defined by a directed acyclic graph on a set of input nodes $x_1, \ldots, x_n$, and computation nodes $u_1, \ldots, u_N$. The input nodes have no incoming edges. One of the computation nodes is called the output node, and has no outgoing edges. Each computation node $u_k$ is associated with a function $f_k : \mathbb{R}^{n_k} \rightarrow \{0, 1\}$, where $n_k$ is $u_k$’s indegree (number of ingoing edges). On input $x \in \mathbb{R}^n$, the network computes its output $g(x)$ in a natural, feedforward fashion. For instance, given input $x = (x_1, x_2, x_3)$, the network above computes $g(x)$ as follows:

$$\begin{align*}
u_1 &= f_1(x_1, x_2, x_3) \\
u_2 &= f_2(x_2, x_3) \\
u_3 &= f_3(u_1, x_2, u_2) \\
u_4 &= f_4(u_1, u_3) \\
g(x) &= u_4.
\end{align*}$$

(Here, we slightly abuse notation, writing $x_j$ and $u_k$ both for nodes of the network, and for the input/computed values associated with these nodes.) The number of edges in the graph is denoted $W$.

In what follows, we regard the underlying graph as fixed, but allow the functions $f_k$ to vary, or to be learned from data. In particular, let $\mathcal{F}_1, \ldots, \mathcal{F}_N$ be spaces of functions. As just explained, every choice of functions $f_1, \ldots, f_N$ induces an overall function $g : \mathbb{R}^n \rightarrow \{0, 1\}$ for the network. We let $\mathcal{G}$ denote the space of all such functions when $f_k$ is chosen from $\mathcal{F}_k$ for $k = 1, \ldots, N$. 


a. Prove that
\[ \Pi_G(m) \leq \prod_{k=1}^{N} \Pi_{F_k}(m). \]
(Note that this is a generalization of Problem 2.)

b. Let \( d_k \) be the VC-dimension of \( F_k \), and let \( d = \sum_{k=1}^{N} d_k \). Assume \( m \geq d_k \geq 1 \) for all \( k \). Prove that
\[ \Pi_G(m) \leq \left( \frac{emN}{d} \right)^d. \]

c. Consider the typical case in which the functions \( f_k \) are linear threshold functions; as we know, this class of functions has VC-dimension \( d_k = n_k + 1 \). Give an exact expression for \( d \) in terms of \( N, n, \) and \( W \). Conclude by deriving a “big-Oh” upper bound on the generalization error of any \( g \in G \) that is consistent with \( m \) random examples, assuming \( m \geq d \). Your bound should hold with probability at least \( 1 - \delta \), and should be expressed in terms of \( N, n, W, m, \) and \( \delta \).