I pleage my honor that I have not violated the Honor Code during this examination.

Name Pranjit Kumar Kalita

Signature [illegible]

COS511 Final

Pranjit Kalita

May 20, 2017

Ex. 1
First of all, we will prove that the VC-dimension of $H \geq (n+1)$. For that we will need to show that a set of size $(n+1)$ is shattered.
Let $S = \{(0,0,...,0),(1,0,...,0),(1,1,0,...,0),...,(1,1,1,...,1,0),(1,1,...,1)\}$, i.e., it represents set of vectors each of size $(n+1)$ elements with each vector having the first i elements as 1, with $i \in [0,n]$.
Now, each element in S has a unique number of 1's, with size of S being $(n+1)$.
Thus, there exists $h \in H$ which is symmetric which gives a unique labeling to every element $\in S$.
Therefore, H shatters S.
$\Rightarrow$VC-dimension of H $\geq (n+1)$.
Now, we will show that VC-dimension of $H < (n+2)$.
Let us consider a set S' of size $(n+2)$ vectors.
Since we have (n + 2) vectors and (n + 1) being the size of each vector, therefore by the Pigeonhole Principle, at least two vectors x_i and $x_j \in S'$, will have the same number of 1's.
Thus although some function $f(x_i) \neq f(x_j)$, for some $h \in H$, $h(x_i) = h(x_j)$ (since no. of 1's is the same).
Thus, H does not shatter S'.
$\Rightarrow$VC-dimension of H < (n + 2).
Thus,VC-dimension of H = (n + 1).
Thus, using Fundamental Theorem of Statistical Learning, since VC-dimension of H is (n + 1),in the agnostic PAC model,
m$(\epsilon,\delta) = O(\frac{(n+1)log(1/\delta)}{\epsilon^2})$
$\Rightarrow$ m$(\epsilon,\delta) = \Theta(\frac{nlog(1/\delta)}{\epsilon^2})$

Ex. 4
The hypothesis class to be learned will be all hypotheses that will return 1 if a sequence of some specific length 'l', $l \in [1, d]$ is found within a message of length 'd', specified by the file which will be our sample data.
Now, since $l \in [1, d]$, and the size of the alphabet is 'a' (let), therefore there will be a^l possible sub-strings that could infect a file.
Thus, total no of possible strings $= \sum_{i=1}^{d} a^l$
$\Rightarrow$No. of strings $= \dfrac{a(a^d - 1)}{a - 1} = O(a^d)$
Since each one of these strings has a corresponding hypothesis, $|H| = \dfrac{a(a^d - 1)}{a - 1}$ $= O(a^d)$
Therefore, H is finite in length and according to Corollary 2.5 (Finite Classes are Learnable), any ERM algorithm can learn this hypothesis class H with sample complexity $m = O(\dfrac{log\dfrac{|H|}{\delta}}{\epsilon^2})$
$\Rightarrow$m = O($\dfrac{d * log\dfrac{a}{\delta}}{\epsilon^2}$)
Now, our domain is $X = A^d$, where A is the alphabet.
In order to embed X into a linear space, given each $h \in H$ (which will be strings that the hypothesis learns as shown above), we will try to embed the no. of times each word (h) appears in each file. Thus, we will have a linear space $V = \mathbb{N}^{|H|}$.
The size of each vector exponential in $|H|$, which in turn is exponential in d, and each coordinate corresponds to for each file/input how many times that word (hypothesis h_i) appears in file. Thus for each coordinate $n_h > 0$, that would be the linear classifier of the corresponding $h \in H$.
In the realizable setting if we were to use the Perceptron algorithm, due to the exponential dependency on $|H|$, it would not yield a very efficient algorithm. In order to counter that, let us use an assumption. Assuming that at least one file f is infected, let us get all the sub-strings within file f, which could be done in $O(\sum_{i=1}^{d}(i)) = O(d^2)$ time (the trick is to find the upper bound of the number of sub-strings within a file of string length d).
Now, the idea is to remove all the sub-strings that do not appear in the files, while knowing those that lead to infected files. Thus, in O(d^2) time we can remove effectively a vast majority of hypotheses from the vector embedding V, and that'd get rid of the exponential dependency on $|H|$. That in turn, and due to the polynomial dependency of sample complexity on d (along with the quadratic dependency on d as shown above), would lead to an efficient algorithm if we were to employ Perceptron Algorithm to learn the linear space classifier V in the realizable setting. Once again, it'd be any $n_h > 0$, with zero training error (since it is realizable).

Ex. 3
We use the hint. We are going to bound the probability for zero error on d sample points but with error $\geq \epsilon$ on all points (m).
$P_S(|err_s(h) - err(h)|) \geq \epsilon$
Since $err(h) = 0$
$\Rightarrow$P$_S(|err_s(h)| \geq \epsilon) \leq 2e^{-2*d*\epsilon^2}$ (Using Claim 2.4 of notes)
In order to apply the union bound, let us find the no of ways in which we can find 'k' subsequences out of 'm' sample points :
$\sum_{k=1}^{d} C(m,k) = O(m^d)$ (By Sauer's Lemma)
The above is the size of the hypothesis class H. Now, $|H| = O(m^d)$
Applying union bound,
$P_S(\exists h \in H|err_S(h) - err(h)| \geq \epsilon) \leq 2|H|e^{-2*d*\epsilon^2}$
Since $err(h) = 0$
$\Rightarrow$P$_S(\exists h \in H|err_S(h)| \geq \epsilon) \leq 2|H|e^{-2*d*\epsilon^2}$ (Using Claim 2.4 of notes)
$\Rightarrow$P$_S(\exists h \in H|err_S(h)| \geq \epsilon) \leq 2 * m^d * e^{-2*d*\epsilon^2}$

Now, proceeding with the proof of Claim 2.4 in notes, if $m = O(\frac{1 * log\frac{2*|H|}{\delta}}{2*\epsilon^2})$ with probability at least $(1-\delta)$
$max_{h\in H}|err_S(h) - err(h)| < \epsilon$
$\Rightarrow$m = O($\frac{d * log\frac{1}{\delta}}{\epsilon^2}$) (since $|H| = m^d$)
Using Theorem 3.5 in notes, $m = O(\frac{d * log\frac{1}{\delta}}{\epsilon^2})$ for ERM learnability of a finite hypothesis class in the agnostic setting.
But since $err(h) = 0$ (according to the hints), since our case is in realizable setting, $m(\epsilon,\delta) = O(\frac{d * log\frac{1}{\delta}}{\epsilon})$
$\Rightarrow$VC-dimension of H is d (with logarithmic factors log($\frac{1}{\delta}$)). (Fundamental Theorem of Statistical Learning) $\Rightarrow$VC$-dim(H) = \widetilde{O}(d)$
Q.E.D.

Ex. 2
We use the given hint. Given a distribution p_t over sample $\{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$ a weak learner returns a hypothesis h_t such that -
$\sum_i p_t(i) \frac{|h_t(i) - y_i|}{2} < \frac{1}{2} - \gamma$
Let us set $g_t(i) = 1 - \frac{|h_t(i) - y_i|}{2}$
Now, $p_t.g_t = \sum_i p_t(i) * (1 - \frac{|h_t(i) - y_i|}{2})$
$\Rightarrow$p$_t.g_t = \sum_i p_t(i) - \sum_i p_t(i) \frac{|h_t(i) - y_i|}{2}$
$\Rightarrow$p$_t.g_t > 1 - (\frac{1}{2} - \gamma)$
$\Rightarrow$p$_t.g_t > \frac{1}{2} + \gamma$
Thus, our selection of $g_t(i)$ yields the correct given conditions A/Q.
To show: $\sum_{t=1}^{T} g_t(i) > \frac{T}{2}$
Now, Since $Regret_T = O(\sqrt{Tlogm})$ for $T > \frac{1}{\gamma^2} logm$
$\Rightarrow \sum_{t=1}^{T} g_t(i) \geq \sum_{t=1}^{T} p_t(i).g_t(i) - Regret_T$
$\Rightarrow \sum_{t=1}^{T} g_t(i) \geq \frac{T}{2} + T\gamma - O(\sqrt{Tlogm})$
$\Rightarrow \sum_{t=1}^{T} g_t(i) > \frac{T}{2} + T\gamma - \sqrt{Tlogm}$ (since $\sqrt{Tlogm} < O(\sqrt{Tlogm})$)
$\Rightarrow \sum_{t=1}^{T} g_t(i) > \frac{\frac{1}{\gamma^2} logm}{2} + \frac{1}{\gamma^2} logm\gamma - \sqrt{\frac{1}{\gamma^2} log^2 m}$
$\Rightarrow \sum_{t=1}^{T} g_t(i) > \frac{\frac{1}{\gamma^2} logm}{2}$
$\Rightarrow \sum_{t=1}^{T} g_t(i) > \frac{T}{2}$
Q.E.D.
For the boosting algorithm which will be online learning according to the error at each iteration g_t, we will begin with a distribution p_t for every iteration, $t \in \{1, T\}$, such that for every h_t that the weak learner generates for the adversary with loss of $g_t(i) = 1 - \frac{|h_t(i) - y_i|}{2}$, the boosting algorithm assigns loss of $g_t(i)$ to the experts. The idea is to assign more weights to high error samples so as the weak learner in the next distribution iteration tries to learn the wrong samples. This is similar to AdaBoost in concept. However, there are also a few differences. AdaBoost assumes Realizable setting whereas this case is agnostic. Also, this online learning assumes convex functions whereas AdaBoost does not.