HW 4

Elwyn Cruz
10497782

1a.) To get an algorithm $A(S_{train})$ w/ 0% training error, our algorithm will look through the training set to find all the labels. There are 2 possible cases: our labels contains an x with a label of 1 or our labels do not contain an x w/ label of 1. (they are all 0.) Then, if there is a label 1, we choose $h^* = h_2$ where z is equal to the x value of the point with label 1. Otherwise, we choose $h^* = h^-$, the hypothesis with all x-input labeled as 0. Either way, our training error will be 0.

1b) $P(L_{(D,h^*)}(A(S_{train})) > \varepsilon) \leq 0$

Loss on training data

Since S_{train} is a random variable, loss is also a random variable

$P(L_{(D,h^*)}(A(S_{train})) > \varepsilon$

$L_{(D,h^*)}(A(S_{train})) > 0$ is a larger event than $L_{(D,h^*)}(A(S_{train})) > \varepsilon$ b/c $0 < \varepsilon$.

So we will look to bound $P(L_{(D,h^*)}(A(S_{train})) > 0)$

want: $\sigma \geq P(L_{(D,h^*)}(A(S_{train})) > 0) \geq P(L_{(D,h^*)}(A(S_{train}) > \varepsilon)$

Each point in the S_{train} has a $1 - \frac{1}{N}$ chance of being the incorrect value ($x \neq z$). Thus for m points, the probability that our $A(S_{train})$ is NOT the real hypothesis h^* is $(1 - \frac{1}{N})^m$

so $\sigma \geq (1 - \frac{1}{N})^m \geq P(L(A(S_{train})) > \varepsilon)$

and we know that $P_{x \sim D}(h^*(x) \neq A(S_{train})(x)) = 1/N$ because for two different hypotheses, the probability they are different is equal to the probability $x = z$ where $W(z) = 1$ and we know $P(x = z) = 1/N$.

so $P_{x \sim D}(h^*(x) \neq A(S_{train})(x)) = 1/N$

If we plug in $m = \frac{\log(1/\sigma)}{\varepsilon}$

$\sigma \geq (1 - \frac{1}{N})^{\frac{\log(1/\sigma)}{\varepsilon}}$

$\log \sigma \geq \frac{\log(1/\sigma)}{\varepsilon} \log(1 - \frac{1}{N})$

$\varepsilon \geq -\log(1 - \frac{1}{N})$

if $(\frac{1}{N} > \varepsilon)$ then $\frac{1}{N} \geq -\log(1 - \frac{1}{N})$ and (generalization error $> \varepsilon$) $-\frac{1}{N} \leq \log(1 - \frac{1}{N})$ →

$-\frac{1}{N} \leq \log(1-\frac{1}{N})$

and $e^{-1/N} \geq 1-\frac{1}{N}$

using the given equation,

we know $1-x \leq e^{-x}$

therefore, using $x=\frac{1}{N}$,

we know $1-\frac{1}{N} \leq e^{-1/N}$,

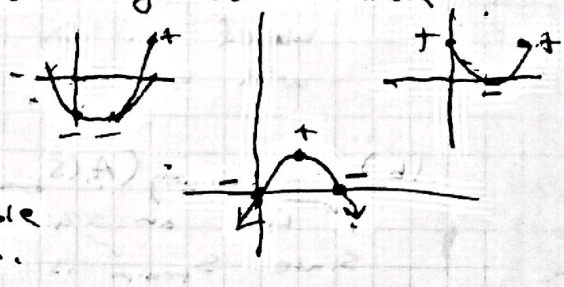
and thus we have bounded

$(1-\frac{1}{N})^M \leq \sigma$,

2.) $H = \{ \text{sgn}(ax^2+bx+c);\ a,b,c \in \mathbb{R}$

given ~~rectangles (0,0), (1,1), (2,0)~~ $x = 0, 1, 2$

we can construct a polynomial function of degree 2 such that the sign at these 3 points can be any combination of + and −.

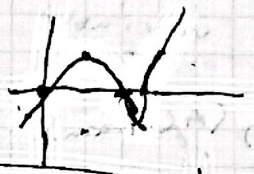


Thus $VC(H) \geq 3$

Given 4 pts, $x = 0, 1, 2, 3$ a combination of (0,0), (1,1), (2,0), (3,1) is impossible and requires a polynomial of degree 3.

Thus $VC(H) < 4$

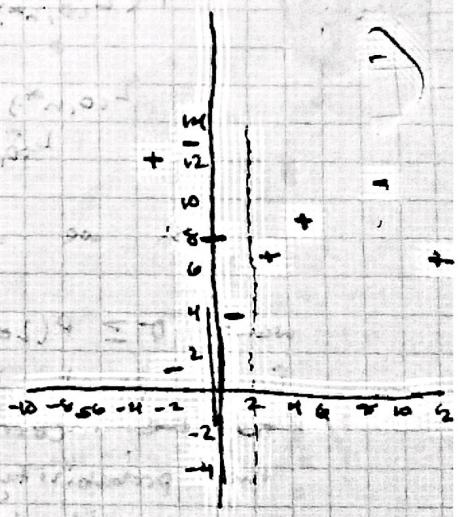
Therefore $VC(H) = 3$



← 3 zeroes = impossible

3.)

		Hypothesis 1				Hypothesis 2			
i	Label	D_0	f_1: $x>2$	f_2: $y>6$	h_1: f_1	D_1	f_1: $x>10$	f_2: $(y>11)$	h_2: f_2
1	−	0.1	−	+ ×	−	.0625	−	−	−
2	−	.1	−	−	−	.0625	−	−	−
3	+	.1	+	+	+	.0625	− ×	− ×	−
4	−	.1	−	−	−	.0625	−	−	−
5	−	.1	−	+ ×	−	.0625	−	+ ×	+
6	−	.1	+ ×	+ ×	+	.25	−	−	−
7	+	.1	+	+	+	.0625	+	− ×	−
8	−	.1	−	−	−	.0625	−	−	−
9	+	.1	− ×	+	−	.25	− ×	+	+
10	+	.1	+	+	+	.0625	− ×	− ×	−
			2/10	3/10					



3c) $\alpha_0 = \frac{1}{2}\log_2\left(\frac{1-.2}{.2}\right) = 1$

$D_1 = \frac{1}{10 z_t} \cdot \begin{cases} e^{-1} & y_i = h_t(x_i) \\ e & \text{otherwise} \end{cases}$

$= \begin{cases} \frac{.05}{z_t} = .0625 \\ \frac{.1}{z_t} = .25 \end{cases}$

$2\left(\frac{.2}{z_t}\right) + 8\left(\frac{.05}{z_t}\right) = 1$

$z_t = .8$

$x > 10$:

$\varepsilon = .25 + 2 \times .0625$

$= 0.375$

$y >$

$\varepsilon = 4 \times .0625 = 0.25$

$\alpha_1 = \frac{1}{2}\log_2\left(\frac{1-.25}{.25}\right) = 0.792$

3d.) $H(x) = \text{sgn}(1 \times [x>2] + 0.79 \times [y>11])$

4a. i) One v All: Learn one classifier for each i, so k classifiers

All v All: Learn one classifier for each $\langle i,j \rangle$ pair.

So for $i=1$, learn $k-1$ classifiers. Thus to learns $C(k,2) = \frac{k(k-1)}{2}$ classifiers

4a. ii) One v All: We use all m examples for each classifier.

All v All: We use $\frac{2m}{k}$ examples for each classifier, m/k from each class.

4a. iii) One v All: For each classifier, determine probability it is that class label v not that label. The one with the highest probability is the final choice.

All vs All: For each classifier, determine the "winner" Take these winners as votes, and determine label with the most votes.

4a iv) One v All: train k classifiers, each using m examples: $O(mk)$

All v All: $\frac{k(k-1)}{2}$ classifiers. $\frac{2m}{k}$ examples $= m(k-1)$

$= O(mk)$

4b.) I prefer one vs all because of its easier implementation and it is more computationally efficient since we are only training k classifiers.

4c.) Yes, using a kernel Perceptron does change my above analysis since kernel perceptron has $O(m^2)$ time

One v All becomes: $O(m^2k)$

All v All becomes: $O(m^2)$

Thus, I would use All v All since it is faster now.

4d.) Using the black box to train each classifier,

a classifier using one vs all would have $O(dm^2)$.

With k classifiers, this becomes $O(kdm^2)$

A classifier using all v all would have $O(d(\frac{2m}{k})^2)$

With $\frac{k(k-1)}{2}$ classifiers, this becomes $O(dm^2)$

Thus all v all becomes more efficient.

4e.) One vs all becomes: $O(kd^2m)$

All vs all becomes: $O(kd^2m)$

The time complexities are the same, so they both have the same efficiency.

4f.) Counting: we run the prediction for each classifier:
Ok This takes $O(k^2)$ time

Knockout: since there are k classes and we eliminate a class each round, the time complexity is $O(k)$.