Advanced Machine Learning

Assignment 2 – June 2023

Tomeh Lara – 507

Q1. (1.25 points) Consider H the following hypothesis class:

$$H = \{h_a : \mathbb{R} \to \{0,1\} \mid a > 0, a \in \mathbb{R}, where\ h_a(x) = 1_{[-a,a]}(x) = \begin{cases} 1, x \in [-a,a] \\ 0, x \notin [-a,a] \end{cases}\}$$

a. Compute the shattering coefficient $\tau_H(m)$ of the growth function for $m \geq 0$ for hypothesis class H. **(1 point)**
b. Compare your result from the previous point with the general upper bound given by the Sauer lemma. Are they equal or different? **(0.25 points)**

Answer Q1.

a. Compute the shattering coefficient $\tau_H(m)$.

From the definition, for a hypothesis class H, $\tau_H(m)$ is the maximum number of different functions from a set C of size m to {0,1} that can be obtained by restricting H to C.

$$\tau_H(m) = \max_{C \subseteq X : |C| = m} |H_C|$$

If VCdim(H) = d then for any m ≤ d we have $\tau_H(m) = 2^m$, H induces all possible functions from C to 0,1.
Sauer-Shelah lemma:
Given VCdim(H) ≤ d ≤, ∞, then for all C ⊂ X s.t. |C| = m > d + 1, we have: $|H_C| \leq (\frac{em}{d})^d$
Sauer-Shelah lemma:
If VC(H)=d, then

$$\tau_H(m) = \begin{cases} 2^m & if\ m \leq d \\ O(m^d) & otherwise \end{cases}$$

Here, h_a is a binary classifier that maps a real number x to either 0 or 1 based on whether x lies within the interval $[-a, a]$ or not. The growth function of the hypothesis class H is defined as the maximum number of distinct dichotomies that can be generated by H over any subset of m points. To compute the shattering coefficient $\tau_H(m)$ of the growth function for $m \geq 0$ for hypothesis class H, we need to calculate the maximum number of dichotomies that can be generated by H over any subset of m points.
Now, let's compute $\tau_H(m)$: To compute the shattering coefficient for the given hypothesis class H, we need to determine the growth function $m_H(m)$, which is the maximum number of dichotomies that H can generate on a set of m points.

For any set of m points, we can consider all possible labelings of the points as either 0 or 1. For H to be able to shatter any set of m points, it must be able to generate all possible 2^m labelings by considering all possible threshold values of a that split the set into two groups.

For m = 1, we have only two possible labelings, (0) and (1), and H is able to generate both by setting a to be any positive value.

For m = 2, we have four possible labelings, (0, 0), (0, 1), (1, 0), and (1, 1). For H to be able to shatter any set of two points, it must be able to generate all possible labelings. We can see that this is not possible, since there is no threshold value of a that can generate the labeling (0, 1) or (1, 0). Therefore, $m_H(2) = 3$.

For m > 2, it is not possible for H to generate all possible labelings, since there will always be some labelings that cannot be generated no matter what value of a is chosen. Therefore, $m_H(m) = 2^m - 1 =$, since we can always generate all labelings except for the one in which all points have label 0.

Therefore, the shattering coefficient of the growth function for the given hypothesis class H is

$\tau_H(m) = 2^m - 1$ for m ≥ 0.

b. Compare your result from the previous point with the general upper bound given by the Sauer lemma.

Next, let's compare this with the upper bound given by the Sauer Lemma:

For the given hypothesis class H, we can compute the VC dimension by finding the largest value of m for which $\tau_H(m) = 2^m$, which is m = 1. Therefore, the VC dimension of H is d = 1, because H can shatter any set of 1 point but not any set of 2 points.

b. The general upper bound for growth functions is:
$\tau_H(m) \leq \sum_{i=0}^{d} C_m^i$, where d = VCdim(H).
In our case, d=1, the upper bound is:

$$\sum_{i=0}^{1} C_m^i = C_m^0 + C_m^1 = \frac{m!}{0!\,m!} + \frac{m!}{1!\,(m-1)!} = 1 + m$$

Since $\tau_H(m) = 2^m - 1$, which grows exponentially with m, it is clear that the Sauer's lemma upper bound is loose and does not provide an exact value for $\tau_H(m)$.

Therefore, the values obtained in the previous part and by using the upper bound provided by the Sauer's Lemma are vastly different. The shattering coefficient obtained directly for the given hypothesis class H is exponential in m while the Sauer's lemma upper bound is bounded by 1.

Hence, the shattering coefficient and the Sauer Lemma bound are different for this hypothesis class.

Q2. (1.5 points) Consider de concept class C_a formed by the union of two closed intervals $[a, a+1] \cup [a+2, a+4]$, where $a \in \mathbb{R}$. Give an efficient ERM algorithm for learning the concept class C_a and compute its complexity for each of the following cases:

a. realizable case. **(1 point)**
b. agnostic case. **(0.5 point)**

Answer Q2.

a. Realizable case.

$$C_2 = \begin{cases} h_{(a,a+1,a+2,a+4)}: \mathbb{R} \to [0,1], & h_{(a,a+1,a+2,a+4)} = 1_{[a,a+1] \cup [a+2,a+4]} \\ a \le a+1 \le a+2 \le a+4, h_{(a,a+1,a+2,a+4)}(x) = \begin{cases} 1, x \in [a, a+1] \cup [a+2, a+4] \\ 0, otherwise \end{cases} \end{cases}$$

Consider $S = \{(x_1, y), (x_2, y_2), \dots, (x_m, y_m)\}$, where $y_i = h^*(x_i), h^* = h_{(a^*,(a+1)^*,(a+2)^*,(a+4)^*)}$.

Consider the following learning algorithm A that takes input S:

i. Sort S in ascending order of x_i.
ii. Go over the sorted training examples and take the intervals where consecutive training examples labeled as positive start and end the intervals. You can obtain one or two intervals (or no interval = there are no positive training examples). In the case that there are no positive examples take $a_s = (a+1)_s = (a+2)_s = (a+4)_s = z$, where z a random point such that $(z, 0)$ doesn't appear in the training set S.
iii. If you obtained just one interval, you can have
$a_s = \min_{\substack{(x_i,y_i) \\ y_i=1}} x_i, (a+1)_s = \max_{\substack{(x_i,y_i) \\ y_i=1}} x_i, (a+2)_s = (a+4)_s = (a+1)_s$
If you obtained two intervals, then
$a_s = \min_{\substack{(x_i,y_i) \\ y_i=1}} x_i, (a+4)_s = \max_{\substack{(x_i,y_i) \\ y_i=1}} x_i, a_s \le (a+1)_s \le (a+2)_s \le (a+4)_s$

Return $h_s = \left(h_{(a_s,(a+1)_s,(a+2)_s,(a+4)_s)}\right) = 1_{[a_s,(a+1)_s] \cup [(a+2)_s,(a+4)_s]}$.

We need to find $m \ge m_{c_2}(\epsilon, \delta)$ such that for $\epsilon > 0, \delta > 0$ and for every $\mathcal{D}$ distribution over $\mathbb{R}$ we have $\underset{S \sim D^m}{P}(\mathcal{L}_{\mathcal{D},h^*}(h_s) > \epsilon) < \delta$

Let $\epsilon > 0, \delta > 0$ and let $\mathcal{D}$ be a distribution over $\mathbb{R}$.

The region where h_s can make errors is always $\subseteq [a^*, (a+4)^*]$.

Case 1: If $\mathcal{D}([a^*, (a+4)^*]) \le \epsilon$, then $\underset{S \sim D^m}{P}\left(\mathcal{L}_{\mathcal{D},h^*}(h_s) > \epsilon\right) = 0$.

Case 2: If $\mathcal{D}([a^*, (a+4)^*]) > \epsilon$

The types of error that h_s can make are:

- False negatives in $[a^*, (a+1)^*]$ and $[(a+2)^*, (a+4)^*]$
- False positives in $((a+1)^*, (a+2)^*)$ if sample S does not contain any points sampled from $((a+1)^*, (a+2)^*)$.

Denote $L_{FP}, L_{FN,1}, L_{FN,2}$ these types of errors, where:

$$L_{FP}(h_s) = \underset{x\sim D}{P}(x \in [a_s, (a+1)_s] \cup [(a+2)_s, (a+4)_s] \backslash ([a^*, (a+1)^*] \cup [(a+2)^*, (a+4)^*]))$$

$$= \underset{x\sim D}{P}(x \in [(a+1)^*, (a+2)^*] \subseteq [a_s, (a+1)_s] \cup [(a+2)_s, (a+4)_s])$$

$$L_{FN,1}(h_s) = \underset{x\sim D}{P}(x \in [a^*, (a+1)^*] \backslash [a_s, (a+1)_s])$$

$$L_{FN,2}(h_s) = \underset{x\sim D}{P}(x \in [(a+2)^*, (a+4)^*] \backslash [(a+2)_s, (a+4)_s])$$

So, if we want to have $\mathcal{L}_{\mathcal{D},h^*}(h_s) > \epsilon$, then one of the numbers $L_{FP}, L_{FN,1}, L_{FN,2}$ must be $> \frac{\epsilon}{3}$. Define:

$$F_1 = \left\{S \sim \mathcal{D}^m \mid L_{FP}(h_s) > \frac{\epsilon}{3}\right\}$$

$$F_2 = \left\{S \sim \mathcal{D}^m \mid L_{FN,1}(h_s) > \frac{\epsilon}{3}\right\}$$

$$F_3 = \left\{S \sim \mathcal{D}^m \mid L_{FN,2}(h_s) > \frac{\epsilon}{3}\right\}$$

So, $\underset{S\sim D^m}{P}\left(\mathcal{L}_{\mathcal{D},h^*}(h_s) \geq \epsilon\right) \leq \underset{S\sim D^m}{P}(F_1 \cup F_2 \cup F_3) \leq \sum_{i=1}^{3} P(F_i)$

$$P(F_1) = \underset{S\sim D^m}{P}\left(\mathcal{L}_{FP}(h_s) > \frac{\epsilon}{3}\right)$$

$=$ (This means that $D([(a+1)^*, (a+2)^*]) > \frac{\epsilon}{3}$ and no point from $[(a+1)^*, (a+2)^*]$ is sampled in S)

$$\leq \left(1 - \frac{\epsilon}{3}\right)^m \leq e^{-\frac{\epsilon}{3}m}$$

$$P(F_2) = \underset{S\sim D^m}{P}\left(\mathcal{L}_{FN,1}(h_s) > \frac{\epsilon}{3}\right)$$

Construct $R_1 = [a^*, a_0]$ and $R_2 = [(a+1)_0, (a+1)^*]$ such that $D(R_1) = D(R_2) = \frac{\epsilon}{6}$.

If $[a_s, (a+1)_s] \cap R_1 \neq \emptyset$ and $[a_s, (a+1)_s] \cap R_2 \neq \emptyset$, then the error made by h_s on $[a^*, (a+1)^*]$ is smaller than $\frac{\epsilon}{6} + \frac{\epsilon}{6} \geq \frac{\epsilon}{3}$.

So, $L_{FN,1}(h_s) > \frac{\epsilon}{3} \Rightarrow [a_s, (a+1)_s] \cap R_1 = \emptyset$ or $[a_s, (a+1)_s] \cap R_2 = \emptyset$.

Define

$F_{21} = \{S \sim D^m \mid [a_s, (a+1)_s] \cap R_1 = \emptyset\}$

$F_{22} = \{S \sim D^m \mid [a_s, (a+1)_s] \cap R_2 = \emptyset\}$

$$P(F_2) \leq P(F_{21} \cup F_{22}) \leq P(F_{21}) + P(F_{22}) = 2 \cdot \left(1 - \frac{\epsilon}{6}\right)^m \leq 2 \cdot e^{-\frac{\epsilon}{6}m}$$

In the same way we can prove that $P(F_3) \leq 2 \cdot e^{-\frac{\epsilon}{6}m}$. So, we obtain that

$$\underset{S \sim D^m}{P}\left(\mathcal{L}_{\mathcal{D},h^*}(h_s) > \epsilon\right) \leq e^{-\frac{\epsilon}{3}m} + 4 \cdot e^{-\frac{\epsilon}{6}m} \leq e^{-\frac{\epsilon}{6}m} + 4 \cdot e^{-\frac{\epsilon}{6}m} = 5 \cdot e^{-\frac{\epsilon}{6}m} < \delta$$

$$\Rightarrow e^{-\frac{\epsilon}{6}m} < \frac{\delta}{5} \mid \cdot \log$$

$$\Rightarrow -\frac{\epsilon}{6}m < \log\frac{\delta}{5} \mid \cdot \left(-\frac{6}{\epsilon}\right)$$

$$m > \frac{6}{\epsilon} \cdot \log\frac{5}{\delta}$$

In the general case, for $C_P =$ reunion of p intervals, the proof is similar, the only differences are that:

There are (p−1) regions of false positives

2p regions of false negatives

So, we have

$$m \geq \frac{2(2p-1)}{\epsilon} \cdot \log\frac{p+2p-1}{\delta}$$

$$m \geq \frac{2(2p-1)}{\epsilon} \cdot \log\frac{3p-1}{\delta}$$

So, the time complexity → given by sorting S= $\mathcal{O}(m \log m)$.

a. Realizable case.

In the realizable case there is a function $h_{a^*,\,(a+1)^*,(a+2)^*,(a+4)^*}(x) = (1_{[a^*,(a+1)^*]} \cup 1_{[(a+2)^*,(a+4)^*]})(x)$ which label the training points

$$S = \{(x_1, y_1), (x_2, y_2), \ldots . (x_m, y_m)\} \qquad y_i = h_{a^*,\,(a+1)^*,(a+2)^*,(a+4)^*}(x_i)$$

we have to sort the points in S in ascending order according to $x's$, finding:

$$S = \left\{\left(x_{\sigma(1)}, y_{\sigma(1)}\right), \left(x_{\sigma(2)}, y_{\sigma(2)}\right), \ldots . \left(x_{\sigma(m)}, y_{\sigma(m)}\right)\right\}, x_{\sigma(1)} \leq x_{\sigma(2)} \leq \cdots \leq x_{\sigma(m)}.$$

Taking into account the ERM algorithm for determining a, (a+1), (a+2), (a+4) after sorting the training set S.

1) If there are only positive examples return $h_{a,a+1,a+2,a+4}$ where $a = x_{\sigma(1)}, (a+1) = (a+2) = (a+4) = x_{\sigma(m)}$.
2) If there are no positive labels return: $h_{a,a+1,a+2,a+4}$ *where* $a = x_{\sigma(1)} - 1, (a+1) = x_{\sigma(1)} - 1,\ (a+2) = x_{\sigma(m)} + 1,\ (a+4) = x_{\sigma(m)} + 1.$
3) If $\exists (x_i, y_i) \in S$ such that $y_i == 1, then$
   $a = (a+1) = min_i = \overline{1,m}\ , y_i = +1^{x_i}$
   $(a+2) = (a+4) = max_i = \overline{1,m}\ , y_i = +1^{x_i}$
4) The final values for at least a and $(a+4)$ are found, now we have to adjust the values for $(a+1)$ and $(a+2)$.. For $i = \overline{2,m}$
   - If $y_{\sigma(i)} == -1\ and\ y_{\sigma(i-1)} == +1$, then $(a+1) = x_{\sigma(i-1)}$
   - If $y_{\sigma(i)} == +1\ and\ y_{\sigma(i-1)} == -1$, then $(a+2) = x_{\sigma(i)}$

   Return $h_{a,a+1,a+2,a+4}$

We have to determine the complexity of this algorithm…

- Sorting $\mathcal{O}(m \cdot log\ m)$.
- Determining there are no positive labels $\mathcal{O}(m)$
- Determining there are only positive labels $\mathcal{O}(m)$
- Final iteration to adjust $(a+1)$ and $(a+2)$, $\mathcal{O}(m)$
  Total complexity $\mathcal{O}(m \cdot log\ m)$.

b. Agnostic case.

In the agnostic case we have various labels for the same point, thus we are dealing with a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$. In the same way to the realizable case, we start by sorting the training set S according to the $x's$, gaining:

$S = \{(x_{\sigma(1)}, y_{\sigma(1)}), (x_{\sigma(2),}y_{\sigma(2)})(x_{\sigma(m),}y_{\sigma(m)})\}$ with $x_{\sigma(1)} \leq x_{\sigma(2)} \leq \cdots \leq x_{\sigma(m)}$

Consider the set Z containing the values of x' with no repetition

$Z = \{z_1, z_2, \dots, z_n\}$

$z_1 = x_{\sigma(1)} < z_2 < \cdots < z_n = x_{\sigma(m)} \quad n \leq m$

1. If all $y_i = 0$, in order to return an ERM algorithm we can pick two intervals outside the training set S, for example $a_s = (a+1)_s = (a+2)_s = (a+4)_s = x_{\sigma(1)} - 1$.
2. Consider all possible two intervals' reunions $Z_{i,j,k,l} = [z_i, z_j] \cup [z_k, z_l], i = \overline{1,n}, j = \overline{i,n}, k = \overline{j,n}, l = \overline{k,n}$.

For the ERM algorithm, we have to determine the solution $Z^* = Z_{i^*,j^*,k^*,l^*}$ with the smallest empirical risk. We calculate this as:

$$Loss(Z_{i,j,k,l}) = \frac{\#\ negative\ points\ inside\ Z_{i,j,k,l} + \#\ positive\ points\ outside\ Z_{i,j,k,l}}{m}$$

First, we pre-compute the total number of positive (pos_prefix_i) and negative points (neg_prefix_i) with value $x \leq x_{\sigma(i)}$ using a dynamic programming approach of prefix-sums. Because we can have multiple points with the same value x, we need the auxiliary pos_i and neg_i, the number of points with positive, respectively negative labels and value $x = x_{\sigma(i)}$. Considering the base case $pos\_prefix_0 = neg\_prefix_0 = 0$, we have the recurrence, for $i = \overline{1,n}$:

- $pos\_prefix_i = pos\_prefix_{i-1} + pos_i$
- $neg\_prefix_i = neg\_prefix_{i-1} + neg_i$

Now, we fix the limits i, j, k, l and find the ones that minimize Loss ($Z_{i,j,k,l}$). An efficient ERM algorithm for this is:

1. Sort S and find $x_{\sigma(1)} \leq x_{\sigma(2)} \leq \cdots \leq x_{\sigma(m)}$. Build set Z containing value x without repetition: $Z = \{z_1, z_2, \ldots, z_n\}$, $z_1 = x_{\sigma(1)} < z_2 < \cdots < z_n = x_{\sigma(m)}$
2. Check if all $y_i\ i = \overline{1,m}$ have value 0. If so, return $h_{(a_s,(a+1)_s,(a+2)_s,(a+4)_s)}$, where $a_s = (a+1)_s = (a+2)_s = (a+4)_s = z_1 - 1$
3. For $j = \overline{1,n}$, calculate values

$$pos_j = \#\ points\ x_i = z_j\ with\ label\ y_i = 1$$
$$neg_j = \#\ \ points\ x_i = z_j\ with\ label\ y_i = 0$$

4. For $i = \overline{1,n}$

$$pos\_prefix_i = pos\_prefix_{i-1} + pos_i$$

$neg\_prefix_i = neg\_prefix_{i-1} + neg_i$

5. min_error $= \frac{m}{m} = 1, i^* = [\,], \; j^* = [\,], k^* = [\,], l^* = [\,]$

   for $i = \overline{1,n}$

   for $j = \overline{i,n}$

   for $k = \overline{j,n}$

   for $l = \overline{k,n}$

$$\text{Loss}\left(Z_{i,j,k,l}\right) = \frac{(neg\_prefix_j - neg\_prefix_{i-1}) + (neg\_prefix_l - neg\_prefix_{k-1})}{m} + \frac{pos\_prefix_n - (pos\_prefix_j - pos\_prefix_{i-1}) - (pos\_prefix_l - pos\_prefix_{k-1})}{m}$$

   If $\text{Loss}\left(Z_{i,j,k,l}\right) < \text{min\_}error$

   min_error = $\text{Loss}\left(Z_{i,j,k,l}\right)$

   $i^* = i$

   $j^* = j$

   $k^* = k$

   $l^* = l$

6. return $h_{(a_s,(a+1)_s,(a+2)_s,(a+4)_s)}$, where $a_s = z_{i^*}, (a+1)_s = z_{j^*}, (a+2)_s = z_{k^*}, (a+4)_s = z_{l^*}$

   Let us calculate the complexity of this algorithm:

   - Sorting $\mathcal{O}(m \cdot log\ m)$.
   - Linear check $\mathcal{O}(m)$
   - Auxiliary counts pre-compute $\mathcal{O}(m)$
   - Prefix-sums DP pre-compute $\mathcal{O}(m)$
   - Finding best i, j, k, l combination $\mathcal{O}(m^4)$ (constant time for Loss using pre-computed prefix-sums)

     Total complexity: $\mathcal{O}(m^4)$.

Q3. (1.25 points) Consider a modified version of the AdaBoost algorithm that runs for exactly three rounds as follows:

- the first two rounds run exactly as in AdaBoost (at round 1 we obtain distribution $D^{(1)}$, weak classifier h_1 with error ϵ_1; at round 2 we obtain distribution $D^{(2)}$, weak classifier h_2 with error ϵ_2).
- in the third round we compute for each $i = 1,2,\dots,m$:

$$D^{(3)}(i) = \begin{cases} \frac{D^{(1)}(i)}{Z}, if\ h_1(x_i) \neq h_2(x_i) \\ 0, \qquad otherwise \end{cases}$$

  where Z is a normalization factor such that $D^{(3)}$ is a probability distribution.

- obtain weak classifier h_3 with error ϵ_3.
- output the final classifier $h_{final}(x) = sign(h_1(x) + h_2(x) + h_3(x))$

  Assume that at each round $t = 1,2,3$ the weak learner returns a weak classifier h_t for which the error ϵ_t satisfies $\epsilon_t \leq \frac{1}{2} - \gamma_t, \gamma_t > 0$.

a. What is the probability that the classifier h_1 (selected at round 1) will be selected again at round 2? Justify your answer. **(0.25 points)**
b. Consider $\gamma = \min\{\gamma_1, \gamma_2, \gamma_3\}$. Show that the training error of the final classifier h_{final} is at most $\frac{1}{2} - \frac{3}{2}\gamma + 2\gamma^3$ and show that this is strictly smaller than $\frac{1}{2} - \gamma$. **(1 point)**

Answer Q3.

a. Step1:

   $D^{(1)}(i) = \frac{1}{m}, \forall i = 1, m$

   h_1 is the is the classifier outputted at step 1. Let t be the number of examples labeled correctly and f be the number of examples labeled incorrectly by h_1, $t + f = m$. Also, let T be the set of examples from the training set that were labeled correctly and F labeled incorrectly. $|T| = t, |F| = f$. We will call the point (x_i, y_i) as p_i for simplicity.
   Then the error of h_1, namely $\epsilon_1 = \sum_{i=1}^{m} D^{(1)}(i) * 1_{[h_1(x_i) \neq y_i]} = \sum_{i=1}^{f} \frac{1}{m} = \frac{f}{m}$

a. Step2:
   Let's suppose that the classifier outputted at step 2 is the same as the one outputted at step 1 ($h_2 = h_1$). We will compute the error of h_1 at step 2, namely ϵ_2.
   $Z_2 = \sum_1^m D^{(1)}(i) * e^{-w_1 h_1(x_i) y_i} = \sum_1^t \frac{1}{m} * e^{-w_1} + \sum_1^f \frac{1}{m} * e^{w_1}$

$$= \frac{t}{m} * \sqrt{\frac{\epsilon_1}{1-\epsilon_1}} + \frac{f}{m} * \sqrt{\frac{1-\epsilon_1}{\epsilon_1}} = \frac{t}{m} * \sqrt{\frac{\frac{f}{m}}{\frac{t}{m}}} + \frac{f}{m} * \sqrt{\frac{\frac{t}{m}}{\frac{f}{m}}} = \frac{2\sqrt{tf}}{m}$$

For $p_i \in T, D_t^{(2)}(i) = \frac{D^{(1)}(i) * e^{-w_1}}{Z_2} = \frac{1}{m} * \frac{\sqrt{\frac{\epsilon_1}{1-\epsilon_1}}}{\frac{2\sqrt{tf}}{m}} = \frac{1}{m} * \sqrt{\frac{\frac{f}{m}}{\frac{t}{m}}} * \frac{m}{2\sqrt{tf}} = \sqrt{\frac{f}{t}} * \frac{1}{2\sqrt{tf}} = \frac{1}{2t}$.

For $p_i \in F, D_f^{(2)}(i) = \frac{D^{(1)}(i) * e^{w_1}}{Z_2} = \frac{1}{m} * \frac{\sqrt{\frac{1-\epsilon_1}{\epsilon_1}}}{\frac{2\sqrt{tf}}{m}} = \frac{1}{m} * \sqrt{\frac{\frac{t}{m}}{\frac{f}{m}}} * \frac{m}{2\sqrt{tf}} = \sqrt{\frac{t}{f}} * \frac{1}{2\sqrt{tf}} = \frac{1}{2f}$.

$$\epsilon_2 = \sum_{i=1}^{m} D^{(2)}(i) * 1_{[h_1(x_i) \neq y_i]} = \sum_{1}^{f} D^{(2)}(i) = f * \frac{1}{2f} = \frac{1}{2}$$

However, the algorithm only outputs weak learners, with $\epsilon \leq \frac{1}{2} - \gamma, \gamma > 0$, so it cannot output h_1 at step 2 as it has $\epsilon = \frac{1}{2}$. Therefore, the probability that the classifier h_1 (selected at round 1) will be selected again at round 2 is 0.

b. Let T be the set of examples from the training set that were labeled correctly by h_1, F the set of examples from the training set that were labeled incorrectly by h_1. TF the set of examples from the training set that were labeled correctly by h_1 but incorrectly by h_2. Similarly, FT is the set of examples from the training set that were labeled incorrectly by $h_1$and correctly by h_2. TT contains examples labeled correctly by both h_1 and h_2 and FF examples labeled incorrectly by both h_1 and h_2. So, we have T, F, TT, TF, FT, FF. For each of them, we will denote the number of elements as follows: for T we have t, for TF we have tf and so on.

At Step 1, we output a classifier h_1 with error $\epsilon_1 = \frac{f}{m} \leq \frac{1}{2} - \gamma_1$.

At Step 2, we output a classifier h_2. The examples that make up it's error are the ones in TF and FF (the ones where it gets the label wrong). So, it's error is $\epsilon_2 = tf * D_t^{(2)}(i) + ff * D_f^{(2)}(i) = tf * \frac{1}{2t} + ff * \frac{1}{2f} = \frac{1}{2} * \left(\frac{tf}{t} + \frac{ff}{f}\right) \leq \frac{1}{2} - \gamma_2$.

At Step 3, the output classifier handles only examples on which the 2 previous classifiers disagree.

We can rewrite it's distribution as follows:

$$D^{(3)}(i) = \begin{cases} \frac{1}{m * Z}, & if\ x \in \{TF, FT\} \\ 0, & if\ x \in \{TT, FF\} \end{cases}$$

We can compute Z because we know that the sum of probabilities of examples has to sum up to 1. So $\frac{tf+ft}{m*Z} = 1 \Rightarrow Z = \frac{tf+ft}{m}$.

Now we have:

$$D^{(3)}(i) = \begin{cases} \frac{1}{tf + ft}, & if\ x \in \{TF, FT\} \\ 0, & if\ x \in \{TT, FF\} \end{cases}$$

Given that the classifier h_3 only labels examples in TF and FT, it's failures will be in the following sets: TFF and FTF (and the examples labeled correctly by it in TFT and FTT).

Therefore, it's error $\epsilon_3 = tff * \frac{1}{tf+ft} + ftf * \frac{1}{tf+ft} = \frac{tff+ftf}{tf+ft} \leq \frac{1}{2} - \gamma_3$.

Now we will look at the training error for the final classifier, h_{final}, let's denoted it as L. The classifier gets the label wrong for an example when either both h_1 and h_2 fail (and h_3 does nothing) or one of the first two classifiers gets the label right, the other gets it wrong and the third one gets it wrong too (he had one job…). So h_{final} fails on TFF ∪ FTF ∪ FF. It's error is L = $\frac{tff+ftf+ff}{m}$.

Now we will try to upper bound this error with $\frac{1}{2} - \frac{3}{2}\gamma + 2\gamma^3$.

L = $\frac{tff+ftf+ff}{m} = \frac{1}{m} * \left(ff + \frac{tff+ftf}{tf+ft} * (tf + ft)\right) = \frac{1}{m} * \left(\frac{ff}{2} + tf * \epsilon_3 + \frac{ff}{2} + ft * \epsilon_3\right) =$

$\frac{1}{m} * \left(\frac{ff}{2} + \frac{tf}{2} + tf\left(\epsilon_3 - \frac{1}{2}\right) + \frac{ff}{2} + \frac{ft}{2} + ft\left(\epsilon_3 - \frac{1}{2}\right)\right) = \frac{ff+tf}{2m} + \frac{ff+ft}{2m} + \frac{1}{m}\left(\epsilon_3 - \frac{1}{2}\right)(tf + ft) \leq \epsilon_2 + \frac{\epsilon_1}{2} + \left(\epsilon_3 - \frac{1}{2}\right)\left(\frac{ft+tf}{m}\right)$.

Some more steps should be here.

Then we upper bound each ϵ_1 as follows: $\epsilon_i \leq \frac{1}{2} - \gamma_i \leq \frac{1}{2} - \gamma$ because γ is the minimum of $\gamma_1, \gamma_2, \gamma_3$.

We then replace each ϵ_i with $\frac{1}{2} - \gamma$ and prove the result. Now we need to prove that:

$\frac{1}{2} - \frac{3}{2}\gamma + 2\gamma^3 < \frac{1}{2} - \gamma \Rightarrow 2\gamma^3 - \frac{1}{2}\gamma < 0 \Rightarrow \gamma^2 - \frac{1}{4} < 0 \Rightarrow (\gamma - \frac{1}{2})(\gamma + \frac{1}{2}) < 0.$

The last inequality is true because $\gamma \in \left(0, \frac{1}{2}\right)$.

Q4. (1 point) Consider H^d_{2DNF} the class of 2-term disjunctive normal form formulae consisting of hypothesis of the form $h : \{0,1\}^d \to \{0,1\}$,

$$h(x) = A_1(x) \vee A_2(x)$$

where $A_i(x)$ is a Boolean conjunction of literals H^d_{conj}.

It is known that the class H^d_{2DNF} is not efficient properly learnable but can be learned improperly considering the class H^d_{2CNF} . Give a γ-weak-learner algorithm for learning the class H^d_{2DNF}which is not a strong PAC learning algorithm for H^d_{2DNF} (like the one considering H^d_{2CNF}). Prove that this algorithm is a γ-weak-learner algorithm for H^d_{2DNF}.

Hint: Find an algorithm that returns $h(x) = 0$ or the disjunction of 2 literals.

Ex-oficio: 0.5 points.

Answer Q4.

We can transform the hypothesis $h(x) = A_1(x) \vee A_2(x)$ into a conjunction of disjunctions of 2 literals by using distributivity of the $\vee$ operation. We have $h(x) = \wedge_{u \in A_1, v \in A_2} (u \vee v)$.

So, a 2 term DNF can be viewed as a conjunction of $(2n)^2$ variables.

$H^n_{3DNF} \subseteq H^{(2n)^2}_{conj}$ with $|H^{(2n)^2}_{conj}| = 3^{(2n)^2} + 1$

Our algorithm that outputs a γ weak learner will return a hypothesis of the form $h'(x) = x_i \vee x_j$,

in other words, it will only use one disjunction of 2 literals instead of a huge conjunction of disjunctions of 2 literals.
