Advanced Machine Learning

Sebastian Cojocariu - Group 407 IA

April 2021

1 Exercise 1

Consider $H = H_1 \cup H_2$, where:
$H_1 = \{h_{\theta_1} : \mathbb{R} \to \{0,1\}, h_{\theta_1}(x) = 1_{[x \geq \theta_1]}(x) = 1_{[\theta_1, +\infty)}(x), \theta_1 \in \mathbb{R}\}$
$H_2 = \{h_{\theta_2} : \mathbb{R} \to \{0,1\}, h_{\theta_2}(x) = 1_{[x < \theta_2]}(x) = 1_{(-\infty, \theta_2)}(x), \theta_2 \in \mathbb{R}\}$

a. Give an efficient ERM algorithm for learning H and compute its complexity for the realizable case.

b. Give an efficient ERM algorithm for learning H and compute its complexity for the agnostic case.

c. Compute the shattering coefficient $\tau_H(m)$ the growth function for $m \geq 0$ for hypothesis class H.

d. Compare your result with the general upper bound for the growth functions and show that $\tau_H(m)$ obtained at previous point c is not equal to the upper bound.

e. Does there exist hypothesis class H for which $\tau_H(m)$ is equal to the general upper bound (over $\mathbb{R}$ or another domain X)? If your answer is yes please provide an example, if your answer is no please provide a justification.

Solution:

a. Consider the set $S = \{(x_1, l_1), ..., (x_m, l_m)\}$ of training samples. Let Positives $= \{x | (x, 1) \in S\}$, Negatives $= \{x | (x, 0) \in S\}$. There are several cases to analyze:

- If Positives $= \emptyset$. Then, choose $h_{\theta_1} \in H_1$ with $\theta_1 = \max(\text{Negatives}) + \epsilon$ or $h_{\theta_2} \in H_2$ with $\theta_2 = \min(\text{Negatives}) - \epsilon$.

- Else if Negatives $= \emptyset$. Choose $h_{\theta_1} \in H_1$ with $\theta_1 = \min(\text{Positives}) - \epsilon$ or $h_{\theta_2} \in H_2$ with $\theta_2 = \max(\text{Positives}) + \epsilon$.

- Else (Positives $\neq \emptyset$, Negatives $\neq \emptyset$). Define:
  $x_{\text{min positive}} = \min(\text{Positives}) \leq x_{\text{max positive}} = \max(\text{Positives})$
  $x_{\text{min negative}} = \min(\text{Negatives}) \leq x_{\text{max negative}} = \max(\text{Negatives})$.
  Since we are in the realizability case, we can only have the following subcases:
  1) $x_{\text{min negative}} > x_{\text{max positive}} \to$ Choose any $\theta_2 \in (x_{\text{max positive}}, x_{\text{min negative}})$, $h_{\theta_2} \in H_2$.
  2) $x_{\text{max negative}} < x_{\text{min positive}} \to$ Choose any $\theta_1 \in (x_{\text{max negative}}, x_{\text{min positive}})$, $h_{\theta_1} \in H_1$.

The overall runtime complexity is $O(m)$, where m is the number of training samples (Just one traversal to find min/max -if exists- for the groups positives/negatives and constant time to compare them). At inference, the overall

complexity would be $O(n)$, where n is the number of the samples received (simply compare each $s \in S$ with the θ found and label them accordingly).
Since VCdim(H) = 2 (see point d) for a detailed proof), we know from The Theorem of statistical Learning that the following inequality holds for PAC learnable: there exists $C_1 > 0, C_2 > 0$ such that $C_1 \frac{d + log(\frac{1}{\delta})}{\epsilon} \leq m_H(\epsilon, \delta) \leq C_2 \frac{d * log(\frac{1}{\epsilon}) + log(\frac{1}{\delta})}{\epsilon}$, where d = VCdim(H) = 2. It follows that the overall complexity $O(m_H(\epsilon, \delta))$ can be written in the form of a polynonial $P(\frac{1}{\epsilon}, \frac{1}{\delta})$, so the algorithm described satisfies ERM's condition.

b. Consider the set $S = \{(x_1, l_1), ..., (x_m, l_m)\}$ of training samples. Let $Z = \text{set}(\{x|(x, l) \in S\}) = \{z_1 < ... < z_n\}, n \leq m$ and Positives $= \{w|(w, 1) \in S\}$, Negatives $= \{w|(w, 0) \in S\}$. Obviously, since we are in the agnostic case, we can have $Positives \bigcap Negatives \neq \emptyset$ There are several cases to analyze:

- If Positives $= \emptyset$. Then, choose $h_{\theta_1} \in H_1$ with $\theta_1 = \max(\text{Negatives}) + \epsilon$ or $h_{\theta_2} \in H_2$ with $\theta_2 = \min(\text{Negatives}) - \epsilon$.

- Else if Negatives $= \emptyset$. Choose $h_{\theta_1} \in H_1$ with $\theta_1 = \min(\text{Positives}) - \epsilon$ or $h_{\theta_2} \in H_2$ with $\theta_2 = \max(\text{Positives}) + \epsilon$.

- Else (Positives $\neq \emptyset$, Negatives $\neq \emptyset$). Define:
  $p_i = |\{j|(w_j, 1) \in S, w_j = z_i, j \in \{1, ..., m\}\}|, i \in \{1, ..., n\}$
  $n_i = |\{j|(w_j, 0) \in S, w_j = z_i, j \in \{1, ..., m\}\}|, i \in \{1, ..., n\}$
  $P = |\{j|(w_j, 1) \in S\}|$
  $A_i = \text{Loss}(h_{\theta_1 = z_i}(S)), i \in \{1, ..., n+1\}$, where $z_{n+1} = z_n + \epsilon$
  $B_i = \text{Loss}(h_{\theta_2 = z_i}(S)), i \in \{1, ..., n+1\}$, where $z_{n+1} = z_n + \epsilon$
  $A_i = A_{i+1} - \frac{p_i}{m} + \frac{n_i}{m}, A_{n+1} = \frac{P}{m}, i \in \{n, n-1, ..., 1\}$
  $B_{i+1} = B_i - \frac{p_i}{m} + \frac{n_i}{m}, B_1 = \frac{P}{m}, i \in \{1, ..., n\}$

- Choose $i \in \text{argmin}A$, $j \in \text{argmin}B$ and return $h_{\theta_1 = z_i}$ if $A_i < B_j$, else return $h_{\theta_2 = z_j}$.

The overall runtime complexity is $O(m * log(m)) + O(2 * m) = O(m * log(m))$, where m is the number of training samples (The first complexity is from sorting Z and the second is from computing p_i, n_i, A_i, B_i and store the indexes with minimum values). At inference, the overall complexity would be $O(n)$, where n is the number of the samples received (simply compare each $s \in S$ with θ found and label them accordingly).
Since VCdim(H) = 2 (see point d) for a detailed proof), we know from The Theorem of statistical Learning that the following inequality holds for agnostic PAC learnable: there exists $C_1 > 0, C_2 > 0$ such that $C_1 \frac{d + log(\frac{1}{\delta})}{\epsilon^2} \leq$

$m_H(\epsilon, \delta) \leq C_2 \dfrac{d + log(\frac{1}{\delta})}{\epsilon^2}$, where $d =$ VCdim(H) $= 2$. Again, it follows that the overall complexity $O(m_H(\epsilon, \delta) * log(m_H(\epsilon, \delta)))$ can be written in the form of a polynonial $P(\frac{1}{\epsilon}, \frac{1}{\delta})$, so the algorithm described satisfies ERM's condition.

c. Fix $m \in \mathbb{N}$ and let $S = \{x_1 < x_2 < ... < x_m\}$ a set of samples of lenght m. The only labels that can be obtained using any $h \in H$ from S are of the form $A = \{0^k 1^{m-k} | k \in \{0, ..., m\}\}$ (using H_1) or $B = \{1^k 0^{m-k} | k \in \{0, ..., m\}\}$ (using H_2). Since $|A| = m + 1, |B| = m + 1$ (as k ranges from 0 to m inclusively and each k uniquely determines a new possible labelling), $|A \bigcap B| = |\{0^m, 1^m\}| = 2$ we have that $|A \bigcup B| = |A| + |B| - |A \bigcap B| = m + 1 + m + 1 - 2 = 2m$. A small proof of why A and B are the only possible labellings is due to the fact that the behaviour of a function $h \in H$ when restricted to S changes only in the region of the points from S, inducing the same labelling for $\theta_1, \theta_2 \in (x_k, x_{k+1}), k \in \{0, ..., m+1\}$, where $x_0 = -\infty, x_{m+1} = +\infty$ on S.

d. We will prove that VCdim(H) $= 2$ and use the the general upper bound for the growth functions: $\tau_H(m) \leq \sum_{i=0}^{\text{VCdim(H)}} \binom{m}{i}$.
To prove that VCdim(H) $= 2$, we need two things:
1. Find a subset $S = \{x_1, x_2\}$ that is shattered by H. For this case choose $S = \{0, 1\}$ (any 2 distinct number would suffice). Then, $\{h_{\theta_1=0}, h_{\theta_1=1}, h_{\theta_2=0}, h_{\theta_2=1}\}$ will give us all the possible labellings (namely, {[1, 1], [0, 1], [0 ,0], [1, 0]}).
2. For any set $S = \{x_1, x_2, x_3\}$, the labellings $[0, 1, 0], [1, 0, 1]$ are not possible (since either on the left or the right of each positive sample, all of them must be positive, depending on which H_1 or H_2 we are extracting h)
Finally, $\tau_H(m) \leq \sum_{i=0}^{\text{VCdim(H)}=2} \binom{m}{i} = 1 + m + \dfrac{m(m-1)}{2}$. Obviously, the right side is different from the $\tau_H(m) = 2m$ we calculated at point c. (one is linear in m while the other is quadratic in m)
Observation $2m \leq 1 + m + \dfrac{m(m-1)}{2} \rightarrow \dfrac{(m-1)(m-2)}{2} \geq 0$. The equality holds only when $m \in \{1, 2\}$, the inequality becoming strict for $m \geq 3$.

e. Choose a hypothesis class H that has VCdim(H) $= +\infty$. For a fixed m, by definition we have that the right side of the upper bound is equal to $\sum_{i=0}^{VCdim(H)=+\infty} \binom{m}{i} = 2^m$ (well-known identity in combinatorics), while the shattering coefficient is equal to $\tau_H(m) = 2^m$ (since H has VCdim(H) $= +\infty$, it can shatter arbitrary large set of points, so it can shatter a set of m points as well, for each m). As an example of such hypothesis class, choose the well-known result in literature that $H = \{f_\theta | f_\theta(x) = \text{sign}(\sin(\theta x))\}$ has VCdim equal to ∞. Alternatively, choose H_1 or H_2 with their definitions from the requirements. For instance, it is very easy to show that has VCdim$(H_1) =$ VCdim$(H_2) = 1$: this is because for a $S = \{0\}$ one can find $\{h_{\theta_1=0}, h_{\theta_1=1}\} \rightarrow \{[1], [0]\}$ or $\{h_{\theta_2=0}, h_{\theta_2=2}\} \rightarrow \{[0], [1]\}$, but no set $S = \{x_1, x_2\}$ can be shattered by H_1 (labelling [1, 0] cannot be obtained) nor H_2 (labelling [0, 1] cannot be obtained). Thus, the upper bound would be $\tau_{H_k}(m) \leq \sum_{i=0}^{\text{VCdim}(H_k)=1} \binom{m}{i} = 1 + m, k \in \{1, 2\}$, which would be an actual equality, using the arguments from point c)

$(H_1 \rightarrow \{0^k 1^{m-k} | k \in \{0, ..., m\}\},\ H_2 \rightarrow \{1^k 0^{m-k} | k \in \{0, ..., m\}\})$ to prove that $\tau_{H_k}(m) = m + 1$.

2 Exercise 2

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, ..., m:
  $D_3(i) = \begin{cases} \dfrac{D_1(i)}{Z} \text{ if } h_1(x_i) \neq h_2(x_i) \\ 0, \text{ otherwise} \end{cases},$
  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 $h_{\text{final}}(x) = \text{sign}(h_1(x) + h_2(x) + h_3(x))$.

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.

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 $\frac{1}{2} - \gamma$.

<u>Solution</u>

a. We will prove that, under the constraint $e_i \leq \frac{1}{2} - \gamma_i, \gamma_i > 0, i \in \{1, 2, 3\}$ there is 0 probability of choosing h_1 in the second round.
We have the following relationship between distributions D_{t+1} and D_t:

- $D_{t+1}(i) = \dfrac{D_t(i) * e^{-w_t h_t(x_i) * y_i}}{Z_{t+1}}$.
- Z_{t+1} is a normalizing factor.
- $w_t = \dfrac{1}{2} * log(\dfrac{1}{\epsilon_t} - 1)$.
- $\epsilon_t = P_{i \sim D_t}[h_t(x_i) \neq y_i] = \Sigma_{h_t(x_i) \neq y_i} D_t(i)$.

- If $h_t(x_i) = y_i \rightarrow D_{t+1}(i) = \dfrac{D_t(i) * e^{-w_t}}{Z_{t+1}} = \dfrac{D_t(i) * e^{-\frac{1}{2}ln(\frac{1}{\epsilon_t}-1)}}{Z_{t+1}} = \dfrac{D_t(i) * \sqrt{\dfrac{\epsilon_t}{1-\epsilon_t}}}{Z_{t+1}}$

- If $h_t(x_i) \neq y_i \rightarrow D_{t+1}(i) = \dfrac{D_t(i) * e^{w_t}}{Z_{t+1}} = \dfrac{D_t(i) * e^{\frac{1}{2}ln(\frac{1}{\epsilon_t}-1)}}{Z_{t+1}} = \dfrac{D_t(i) * \sqrt{\dfrac{1-\epsilon_t}{\epsilon_t}}}{Z_{t+1}}$

- $Z_{t+1} = \Sigma_{h_t(x_i)=y_i} D_t(i)\sqrt{\dfrac{\epsilon_t}{1-\epsilon_t}} + \Sigma_{h_t(x_i)\neq y_i} D_t(i)\sqrt{\dfrac{1-\epsilon_t}{\epsilon_t}} = (1-\epsilon_t) * \sqrt{\dfrac{\epsilon_t}{1-\epsilon_t}} + \epsilon_t * \sqrt{\dfrac{1-\epsilon_t}{\epsilon_t}} = 2 * \sqrt{\epsilon_t * (1-\epsilon_t)}$

Suppose that $h_2 = h_1$.

Then, we have that $\epsilon_2 = \Sigma_{h_2(x_i)\neq y_i} D_2(i) = \Sigma_{h_1(x_i)\neq y_i} D_2(i) = \Sigma_{h_1(x_i)\neq y_i} \dfrac{D_1(i) * \sqrt{\dfrac{1-\epsilon_1}{\epsilon_1}}}{Z_2} = \dfrac{\Sigma_{h_1(x_i)\neq y_i} D_1(i) * \sqrt{\dfrac{1-\epsilon_1}{\epsilon_1}}}{2 * \sqrt{\epsilon_1 * (1-\epsilon_1)}} = \dfrac{\sqrt{\dfrac{1-\epsilon_1}{\epsilon_1}} * \Sigma_{h_1(x_i)\neq y_i} D_1(i) *}{2 * \sqrt{\epsilon_1 * (1-\epsilon_1)}} = \dfrac{\sqrt{\dfrac{1-\epsilon_1}{\epsilon_1}} * \epsilon_1}{2 * \sqrt{\epsilon_1 * (1-\epsilon_1)}} = \dfrac{1}{2}$. But from hypothesis we have that $\epsilon_2 = \dfrac{1}{2} \leq \dfrac{1}{2} - \gamma_2 \rightarrow \gamma_2 \leq 0$, a contradiction with $\gamma_i > 0, i \in \{1, 2, 3\}$.

b. We will first prove that: $D_{t+1}(i) = \dfrac{D_t(i)}{1 + y_i h_t(x_i) * (1 - 2 * \epsilon_t)}$. This can be proved by considering $D_{t+1}(i) = \dfrac{D_t(i) * e^{-w_t h_t(x_i) * y_i}}{Z_{t+1}} = \dfrac{D_t(i) * e^{-w_t h_t(x_i) * y_i}}{2 * \sqrt{\epsilon_t * (1-\epsilon_t)}}$. But $w_t = \dfrac{1}{2} * log(\dfrac{1}{\epsilon_t} - 1) \rightarrow e^{-w_t} = \sqrt{\dfrac{\epsilon_t}{1-\epsilon_t}}$, so it follows that $D_{t+1}(i) = \dfrac{D_t(i) * e^{-w_t h_t(x_i) * y_i}}{2 * \sqrt{\epsilon_t * (1-\epsilon_t)}} = \dfrac{D_t(i) * \sqrt{\dfrac{\epsilon_t}{1-\epsilon_t}}^{h_t(x_i) * y_i}}{2 * \sqrt{\epsilon_t * (1-\epsilon_t)}}$. It is very easy to see that for $h_t(x_i) * y_i \in \{-1, 1\}$(which are the only possible combinations) we get the same results in the two relations, so they are indeed equals.

Let $a = P_{i\sim D_2}[h_1(x_i) \neq y_i$ and $h_2(x_i) \neq y_i]$.

We will calculate the following probabilities:
a. $A = P_{i\sim D_1}[h_1(x_i) \neq y_i$ and $h_2(x_i) \neq y_i]$.
b. $B = P_{i\sim D_1}[h_1(x_i) \neq y_i$ and $h_2(x_i) = y_i]$.
c. $C = P_{i\sim D_1}[h_1(x_i) = y_i$ and $h_2(x_i) \neq y_i]$.

d. $D = P_{i\sim D_1}[h_1(x_i) = y_i \text{ and } h_2(x_i) = y_i]$.
e. $E = P_{i\sim D_1}[h_1(x_i) \neq h_2(x_i) \text{ and } h_3(x_i) \neq y_i]$.
f. $F = P_{i\sim D_1}[H(x_i) \neq y_i]$.

Proof:

a. $A = P_{i\sim D_1}[h_1(x_i) \neq y_i \text{ and } h_2(x_i) \neq y_i] = \Sigma_{h_1(x_i)\neq y_i \text{ and } h_2(x_i)\neq y_i} D_1(i) = \Sigma_{h_1(x_i)\neq y_i \text{ and } h_2(x_i)\neq y_i} D_2(i)(1+y_i h_1(x_i)*(1-2*\epsilon_1)) = \Sigma_{h_1(x_i)\neq y_i \text{ and } h_2(x_i)\neq y_i} D_2(i)(1+(-1)*(1-2*\epsilon_1)) = \Sigma_{h_1(x_i)\neq y_i \text{ and } h_2(x_i)\neq y_i} D_2(i)*2*\epsilon_1 = 2*\epsilon_1*\Sigma_{h_1(x_i)\neq y_i \text{ and } h_2(x_i)\neq y_i} D_2(i) = 2*\epsilon_1*P_{i\sim D_2}[h_1(x_i) \neq y_i \text{ and } h_2(x_i) \neq y_i] = 2*a*\epsilon_1$.

b. Since $\epsilon_1 = P_{i\sim D_1}[h_1(x_i) \neq y_i] = P_{i\sim D_1}[h_1(x_i) \neq y_i \text{ and } h_2(x_i) \neq y_i] + P_{i\sim D_1}[h_1(x_i) \neq y_i \text{ and } h_2(x_i) = y_i] = A+B \rightarrow B = P_{i\sim D_1}[h_1(x_i) \neq y_i \text{ and } h_2(x_i) = y_i] = \epsilon_1 - A = \epsilon_1 * (1-2*a)$.

c. $C = P_{i\sim D_1}[h_1(x_i) = y_i \text{ and } h_2(x_i) \neq y_i] = \Sigma_{h_1(x_i)=y_i \text{ and } h_2(x_i)\neq y_i} D_1(i) = \Sigma_{h_1(x_i)=y_i \text{ and } h_2(x_i)\neq y_i} D_2(i)(1+y_i h_1(x_i)*(1-2*\epsilon_1) = \Sigma_{h_1(x_i)=y_i \text{ and } h_2(x_i)\neq y_i} D_2(i)(1+1*(1-2*\epsilon_1) = \Sigma_{h_1(x_i)=y_i \text{ and } h_2(x_i)\neq y_i} D_2(i)*2*(1-\epsilon_1) = 2*(1-\epsilon_1)*\Sigma_{h_1(x_i)=y_i \text{ and } h_2(x_i)\neq y_i} D_2(i) = 2*(1-\epsilon_1)*P_{i\sim D_2}[h_1(x_i) = y_i \text{ and } h_2(x_i) \neq y_i]$.
But, we know that $\epsilon_2 = P_{i\sim D_2}[h_2(x_i) \neq y_i] = P_{i\sim D_2}[h_1(x_i) \neq y_i \text{ and } h_2(x_i) \neq y_i] + P_{i\sim D_2}[h_1(x_i) = y_i \text{ and } h_2(x_i) \neq y_i]$, so it follows that $C = 2*(1-\epsilon_1)*(\epsilon_2 - a)$.

d. We know that $1-\epsilon_1 = P_{i\sim D_1}[h_1(x_i) = y_i] = P_{i\sim D_1}[h_1(x_i) = y_i \text{ and } h_2(x_i) \neq y_i] + P_{i\sim D_1}[h_1(x_i) = y_i \text{ and } h_2(x_i) = y_i] = C + D \rightarrow D = 1 - \epsilon_1 - C$.

e. Additionally, from $\epsilon_3 = P_{i\sim D_3}[h_3(x_i) \neq y_i] = \Sigma_{h_3(x_i)\neq y_i} D_3(i) = \Sigma_{h_3(x_i)\neq y_i \text{ and } h_1(x_i)\neq h_2(x_i)} \frac{D_1(i)}{Z} + \Sigma_{h_3(x_i)\neq y_i \text{ and } h_1(x_i)=h_2(x_i)} 0 = \Sigma_{h_3(x_i)\neq y_i \text{ and } h_1(x_i)\neq h_2(x_i)} \frac{D_1(i)}{Z} = \frac{E}{Z} \rightarrow E = Z*\epsilon_3$.
Now, $Z = \Sigma_{h_1(x_i)\neq h_2(x_i)} D_1(i) = P_{i\sim D_1}[h_1(x_i) \neq y_i \text{ and } h_2(x_i) = y_i] + P_{i\sim D_1}[h_1(x_i) = y_i \text{ and } h_2(x_i) \neq y_i] = B + C$. This implies that $E = \epsilon_3 * (B+C) = -2*a*\epsilon_3 + \epsilon_1*\epsilon_3 + 2*\epsilon_2*\epsilon_3 - 2*\epsilon_1*\epsilon_2*\epsilon_3$.

f. $F = P_{i\sim D_1}[H(x_i) \neq y_i] = P_{i\sim D_1}[h_1(x_i) \neq h_2(x_i) \text{ and } h_3(x_i) \neq y_i] + P_{i\sim D_1}[h_1(x_i) \neq y_i \text{ and } h_2(x_i) \neq y_i] = A + E = 2*a*\epsilon_1 + \epsilon_3[\epsilon_1*(1-2*a) + 2*(1-\epsilon_1)*(\epsilon_2 - a)] = 2*a*(\epsilon_1 - \epsilon_3) + \epsilon_1*\epsilon_3 + 2*\epsilon_2*\epsilon_3 - 2*\epsilon_1*\epsilon_2*\epsilon_3$. From c. we have that $\epsilon_2 \geq a$ (otherwise $C < 0$, which would be absurd for a probability). If we prove that $F \leq 3*\epsilon_{\max}^2 - 2*\epsilon_{\max}^3$, where $\epsilon_{\max} = \max(\epsilon_1, \epsilon_2, \epsilon_3)$, then we can use the fact that $\epsilon_{\max} \leq \frac{1}{2} - \gamma_{\min}$ (since $\epsilon_i \leq \frac{1}{2} - \gamma_i \leq \frac{1}{2} - \gamma_{\min}, i \in \{1,2,3\}$) to get the desired conclusion. Depending on the sign of $\epsilon_1 - \epsilon_3$, we have that either:

- $F \leq \epsilon_1*\epsilon_3 + 2*\epsilon_2*\epsilon_3 - 2*\epsilon_1*\epsilon_2*\epsilon_3$ (when $\epsilon_1 \leq \epsilon_3$).
- $F \leq 2*\epsilon_2*(\epsilon_1 - \epsilon_3) + \epsilon_1*\epsilon_3 + 2*\epsilon_2*\epsilon_3 - 2*\epsilon_1*\epsilon_2*\epsilon_3 = \epsilon_1*\epsilon_3 + 2*$

$\epsilon_1 * \epsilon_2 - 2 * \epsilon_1 * \epsilon_2 * \epsilon_3$ (when $\epsilon_1 > \epsilon_3$).

We will only prove for the first case, the other being extremely similar(it just an interchange of variables for ϵ_i). So, we have to prove that $\epsilon_1 * \epsilon_3 + 2 * \epsilon_2 * \epsilon_3 - 2 * \epsilon_1 * \epsilon_2 * \epsilon_3 = \epsilon_3 * (\epsilon_1 + 2 * \epsilon_2 - 2\epsilon_1 * \epsilon_2) \leq 3 * \epsilon_{\max}^2 - 2 * \epsilon_{\max}^3 = \epsilon_{\max} * (3 * \epsilon_{\max} - 2 * \epsilon_{\max}^2)$. Of course, since $\epsilon_i \leq \epsilon_{\max}, i \in \{1, 2, 3\}$ obviously holds, and the quantity $(\epsilon_1 + 2 * \epsilon_2 - 2\epsilon_1 * \epsilon_2)$ is obviously positive (considering that $0 \leq \epsilon_i < \frac{1}{2}, i \in \{1, 2, 3\}$ and notice that $2 * \epsilon_2 \geq 2 * \epsilon_1 * \epsilon_2$), it would be enough to prove that $(\epsilon_1 + 2 * \epsilon_2 - 2\epsilon_1 * \epsilon_2) \leq 3 * \epsilon_{\max} - 2 * \epsilon_{\max}^2$. Since $\epsilon_{\max} = \max(\epsilon_1, \epsilon_2, \epsilon_3)$, there exists $r_i \geq 0, i \in \{1, 2, 3\}$ such that $\epsilon_i = \epsilon_{\max} - r_i, i \in \{1, 2, 3\}$. The left expression becomes: $\epsilon_1 + 2 * \epsilon_2 - 2\epsilon_1 * \epsilon_2 = (\epsilon_{\max} - r_1) + 2 * (\epsilon_{\max} - r_2) - 2 * (\epsilon_{\max} - r_1) * (\epsilon_{\max} - r_2) = 3 * \epsilon_{\max} - 2 * \epsilon_{\max}^2 - X$, where $X = r_1 + 2 * r_2 - 2 * \epsilon_{\max} * (r_1 + r_2) + 2 * r_1 * r_2$. But $\epsilon_{\max} < \frac{1}{2}$ (hypothesis), so $X \geq r_1 + 2 * r_2 - 2 * \frac{1}{2} * (r_1 + r_2) + 2 * r_1 * r_2 = r_2 + 2 * r_1 * r_2 \geq 0$, so the desired inequality is proved.
So far, we proved that $P_{i \sim D_1}[H(x_i) \neq y_i] \leq 3 * \epsilon_{\max}^2 - 2 * \epsilon_{\max}^3$. But the function $f(x) = 3 * x^2 - 2 * x^3$, which has $f'(x) = 6 * x - 6 * x^2 = 6 * x * (1 - x) > 0$ for $x \in (0, \frac{1}{2})$ is strictly increasing, so we have that $P_{i \sim D_1}[H(x_i) \neq y_i] \leq 3 * \epsilon_{\max}^2 - 2 * \epsilon_{\max}^3 = f(\epsilon_{\max}) \leq f(\frac{1}{2} - \gamma_{\min}) = \frac{1}{2} - \frac{3}{2} * \gamma_{\min} + 2 * \gamma_{\min}^3$, as required.
To prove that $\frac{1}{2} - \frac{3}{2} * \gamma_{\min} + 2 * \gamma_{\min}^3 < \frac{1}{2} - \gamma_{\min}$ note that this is equivalent with (after reducing the terms): $\frac{1}{2} * \gamma_{\min} > 2 * \gamma_{\min}^3$, or $\gamma_{\min} < \frac{1}{2}$, which is obvious (from $0 < \epsilon_i \leq \frac{1}{2} - \gamma_i \leq \frac{1}{2} - \gamma_{\min} \rightarrow \gamma_{\min} \in (0, \frac{1}{2})$).
Observation: Since $\gamma_i > 0, i \in \{1, 2, 3\} \rightarrow \gamma_{\min} = \min_{i \in \{1,2,3\}} \gamma_i > 0$.

3 Exercise 3

Let Σ be a finite alphabet and let $X = \Sigma^m$ be a sample space of all strings of length m over Σ. Let H be a hypothesis space over X, where $H = \{h_w : \Sigma^m \rightarrow \{0, 1\}, w \in \Sigma^*, 0 < |w| \leq m,$ such that $h_w(x) = 1$ if w is a substring of x$\}$

a. Give an upper bound (any upper bound that you can come up) of the VC-dimension of H in terms of Σ and m.

b. Give an efficient algorithm for finding a hypothesis h_w consistent with a training set in the realizable case. What is the complexity of your algorithm?

Solution:

Lemma: If H is a finite hypothesis class, then $\text{VCdim}(H) \leq \lceil \log_2(|H|) \rceil$.
Proof: Suppose $\text{VCdim}(H) = d$. Then, there exist a set of d samples $\{x_1, ..., x_d\}$ that is shattered by H. This, in turn, implies that there exists 2^d labellings of these points that can be achieved using classifiers from H. Since each such

classifier must be different for each labelling, if follows that $|H| \geq 2^d \rightarrow d \leq \lceil \log_2(|H|) \rceil$

a. The key idea is that H is finite (fixing m and Σ). Moreover, $|H| = \text{card}(\{w | w \in \Sigma \cup \Sigma^2 \cup ... \cup \Sigma^m\})$ (the set of all words of length less or equal than m that can be formed using letters from Σ). Obviously, $|\Sigma^k| = |\Sigma|^k$ (there are k positions and for each position one can choose $|\Sigma|$ letters. The conclusion follows by the cartesian product rule). Using the lemma, we have that $\text{VCdim}(H) \leq \lceil \log_2(|\Sigma| + ... + |\Sigma|^m) \rceil = \lceil \log_2(\frac{|\Sigma|^{m+1} - 1}{|\Sigma| - 1} - 1) \rceil$ (if $|\Sigma| = 1$, consider the first part of the equality as the second does not hold).

b. An informal description of the algorithm A: For each word from the given set we compute every substring possible and store them into their own look-up table (duplicates are stored only once). Next, we group the words based on their label and create two substrings sets; call them $\text{Substrings}_{\text{negative}}$, $\text{Substrings}_{\text{positive}}$. For $\text{Substrings}_{\text{positive}}$ we choose only the substrings that appear in all the look-up tables of the positive words (intersection), while for $\text{Substrings}_{\text{negative}}$ we choose all of them (reunion). The final idea is to choose the biggest substring (based on length) $s \in \text{Substrings}_{\text{positive}} \setminus \text{Substrings}_{\text{negative}}$. Since we are in the realizability case, we know that such s must exists (if there is at least one positive word). If there is no positive word, choose any word $w \in \Sigma^{\leq m} \setminus \text{Substrings}_{\text{negative}}$.

Formally, let the training set $S = \{(w_1, l_1), ..., (w_n, l_n)\}$ and consider following algorithm:

- For each $(w, l) \in S$, compute substrings_w(duplicates will be stored only once)(1)
- $\text{Substrings}_{\text{positive}} = \bigcap_{(w,1) \in S} \text{substrings}_w$
  $\text{Substrings}_{\text{negative}} = \bigcup_{(w,0) \in S} \text{substrings}_w$(2)
- If $\text{Substrings}_{\text{positive}} \setminus \text{Substrings}_{\text{negative}} \neq \emptyset$, choose the biggest w from this difference-set based on length (if there are multiple, choose any of them). Otherwise, choose any $w \in \Sigma^{\leq m} \setminus \text{Substrings}_{\text{negative}}$. (Realizability case assures us that such w must exist) (3)
- Return h_w (4)

Complexity analysis:
(1) For a word w for length m there are $\Sigma_{i=1}^m \Sigma_{j=i}^m 1 = \Sigma_{i=1}^m (m - i + 1) = m^2 - \frac{m(m+1)}{2} + m = \frac{m(m+1)}{2}$.
Suppose the hash function used by a hashtable has a runtime of $O(k)$ operations to compute a hash for a word of length k. The complexity to compute the hashes for all the substrings of a word of length m is $\Sigma_{i=1}^m \Sigma_{j=i}^m$(compute hash for substring i...j) $=$ $\Sigma_{i=1}^m \Sigma_{j=i}^m (j - i + 1) = \Sigma_{i=1}^m ((\Sigma_{j=i}^m j) - (i-1) * (m - i + 1)) = \Sigma_{i=1}^m (\frac{m(m+1)}{2} - \frac{(i-1)i}{2} - (i-1)(m-i+1)) = \frac{m^2(m+1)}{2} - \Sigma_{i=1}^m \frac{(i-1)(2m-i+2)}{2} = \frac{m^2(m+1)}{2} -$

$\Sigma_{i=1}^{m}\dfrac{-i^2+i*(2m+3)-(2m+2)}{2}=\dfrac{m^2(m+1)}{2}+\dfrac{m(m+1)(2m+1)}{12}-\dfrac{m(m+1)(2m+3)}{4}+m(m+1)=\dfrac{m(m+1)(m+2)}{6}$ (We used the well known identities: $\Sigma_{i=1}^{m}i=\dfrac{m(m+1)}{2},\Sigma_{i=1}^{m}i^2=\dfrac{m(m+1)(2m+1)}{6}$). So, in order to create such a table, we need $O(\dfrac{m(m+1)(2m+1)}{6})+O(\dfrac{m*(m+1)}{2})=O(m^3)$ (first is from computing the hashes for each substring and the second is from inserting of each substring in the look-up table one the hash is computed; assuming that check existence/delete/insert is $O(1)$ amortized once the hashes are computed). So Step 1 takes $O(nm^2)$ or $O(nm^3)$ (depending on whether computing the hash is $O(1)$ or $O(k)$), where n is the number of training samples.

(2) For step two, we can use two hashtables/dictionaries to find $\text{Substrings}_{\text{positive}}$ and $\text{Substrings}_{\text{negative}}$.

- For $\text{Substrings}_{\text{positive}}$, simply initialize a dictionary to be the first look-up table of the first positive word (if any). Next, for each substring in this dictionary, iterate and check if all the other look-up tables contain it (corresponding to the remaining positive words from the training set). If there is at least one that does not contain it, remove it from this dictionary and go to the next substring from the dictionary. Return dictionary (possible modified). The complexity is at most: $O(\dfrac{nm(m+1)}{2})=O(nm^2)(\dfrac{m(m+1)}{2}$ coming from all the possible substrings of a word of length m, while n comes from checking whether each such substring is part of the other positive look-up tables, which are at most $n-1$) (or $O(nm^3)$ is we suppose again computing the hash is $O(k)$ for a word of length k).
- For $\text{Substrings}_{\text{negative}}$, simply aggregate all the look-up tables of the negative samples: there are at most n such look-up tables of at most $\dfrac{m(m+1)}{2}$ substrings each, so the overall complexity is at most $O(nm^2)$ or $O(nm^3)$ (depending on whether computing the hash is $O(1)$ or $O(k)$)

Overall complexity for Step 2 is $O(nm^2)$ or $O(nm^3)$ (depending on whether computing the hash is $O(1)$ or $O(k)$)

(3), Take into consideration that $|\text{Substrings}_{\text{positive}}|\leq\dfrac{m(m+1)}{2}$,$|\text{Substrings}_{\text{positive}}|\leq\dfrac{nm(m+1)}{2}$. Computing the largest $s\in\text{Substrings}_{\text{positive}}\setminus\text{Substrings}_{\text{negative}}$ (if exists), will take at most $O(|\text{Substrings}_{\text{positive}}|)$ or $O(|\text{Substrings}_{\text{positive}}*m|)$ (if we assume again that computing a hash for a word of length k takes $O(k)$). Choosing a $w\in\Sigma^{\leq m}\setminus\text{Substrings}_{\text{negative}}$ will take at most $O(nm^2)$ or $O(nm^3)$(try the first $\dfrac{nm(m+1)}{2}+1$ words from $\Sigma^{\leq m}$ and stop when you find one that is not in $\text{Substrings}_{\text{negative}}$). Consequently, the overall complexity for this step is $O(nm^2)$ or $O(nm^3)$ (depending whether computing the hash is $O(1)$ or $O(k)$).

(4) This step takes $O(1)$.
Final complexity:

- $O(1)$ for computing a hash for a word of length k: $O(nm^2) + O(nm^2) + O(nm^2) + O(1) = O(nm^2)$.
- $O(k)$ for computing a hash for a word of length k: $O(nm^3) + O(nm^3) + O(nm^3) + O(1) = O(nm^3)$.

Since $|\text{VCdim(H)}| < \infty$ we know from The Theorem of statistical Learning that the following inequality holds for PAC learnable: there exists $C_1 > 0, C_2 > 0$ such that $C_1 \dfrac{d + log(\frac{1}{\delta})}{\epsilon} \leq m_H(\epsilon, \delta) \leq C_2 \dfrac{d * log(\frac{1}{\epsilon}) + log(\frac{1}{\delta})}{\epsilon}$, where $d = \text{VCdim(H)}$. It follows that the overall complexity $O(m_H(\epsilon, \delta) * m^2)$ or $O(m_H(\epsilon, \delta) * m^3)$ (depending on whether computing the hash is $O(1)$ or $O(k)$)) can be written in the form of a polynomial $P(\frac{1}{\epsilon}, \frac{1}{\delta})$, so the algorithm described satisfies ERM's condition (since m and d here are constants).
At inference, for a sample set of size n, the complexity will be $O(nm)$ ($O(m)$ is from the KMP algorithm to check the existance of a given substring in a larger string and $O(n)$ is from iterating over all samples).