Advanced Machine Learning - Assignment 2

Adrian Iordache - Artificial Intelligence

October 10, 2021

Exercise 1

Statement:

Consider $\mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2$ where

$\mathcal{H}_1 = \{h_{\theta_1} : \mathbb{R} \longrightarrow \{0,1\},\ h_{\theta_1}(x) = \mathbf{1}_{[x \geq \theta_1]}(x) = \mathbf{1}_{[\theta_1, +\infty)}(x),\ \theta_1 \in \mathbb{R}\}$ and

$\mathcal{H}_2 = \{h_{\theta_2} : \mathbb{R} \longrightarrow \{0,1\},\ h_{\theta_2}(x) = \mathbf{1}_{[x < \theta_2]}(x) = \mathbf{1}_{(-\infty, \theta_2)}(x),\ \theta_2 \in \mathbb{R}\}$

a) Give an efficient ERM algorithm for learning $\mathcal{H}$ and compute its complexity for the realizable case.

b) Give an efficient ERM algorithm for learning $\mathcal{H}$ and compute its complexity for the agnostic case.

c) Compute the shattering coefficient $\tau_H(m)$ of the growth function for m ≥ 0 for hypothesis class $\mathcal{H}$.

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

Solution:

Point (a): We are in the realizable case $\Longrightarrow$

Given a training set $S = \{(x_1, y_1), (x_2, y_2), \dots (x_m, y_m) \mid x_i \in \mathbb{R}, y_i \in \{0,1\}, i = \overline{1, m}\}$ exists a function $h_{\theta^*} \in \mathcal{H}$ such that $y_i = h_{\theta^*}(x_i)$.

We consider the following scenarios:

1) $- -$ (only negative examples)

2) $+ +$ (only positive examples)

3) $- - - - - - - - - - \theta_1^* + + + + + + + + + +$ (case when $h_\theta^* \in \mathcal{H}_1$)

4) $+ + + + + + + + + + \theta_2^* - - - - - - - - - -$ (case when $h_\theta^* \in \mathcal{H}_2$)

Based on the previous scenarios we can consider the following algorithm:

Input: S
Output: $h^s_{\theta_1}$ or $h^s_{\theta_2}$
Initialization step: $\theta_1 = +\infty$; $\theta_2 = -\infty$;
$\alpha = \underset{i=\overline{1,m}}{argmin}\ x_i$; `// The index of the first sample in S`
`// Checking the label of the first sample to choose a hypotesis class (`$\mathcal{H}_1$ *or* $\mathcal{H}_2$`)`
if $y_\alpha == 0$ **then**
    `// Set as threshold as the first positive example`
    $\theta_1 = \min\limits_{\substack{i=\overline{1,m} \\ y_i=1}} x_i$ if there is such x_i
    return h_{θ_1}
if $y_\alpha == 1$ **then**
    `// Set as threshold the last positive example`
    $\theta_2 = \max\limits_{\substack{i=\overline{1,m} \\ y_i=1}} x_i$
    return h_{θ_2}

Complexity:

- Initialization step: $O(m)$
- Finding θ_1 or θ_2: $O(m)$

Total Complexity: $O(m)$.

Point (b): We are under the agnostic case $\Longrightarrow$ it might not be a function h_{θ} that labels the data (same point might have different labels) or if h_{θ} exists, it might not be in $\mathcal{H}$.

Let's consider training set $S = \{(x_1, y_1), (x_2, y_2), \dots (x_m, y_m) \mid x_i \in \mathbb{R}, y_i \in \{0, 1\}, i = \overline{1, m}\}$. Based on the fact that we are under agnostic case, we can have $x_i = x_j$ and $y_i \neq y_j$.

We consider the following algorithm:

Input: S
Output: $h^s_{\theta_1}$ or $h^s_{\theta_2}$
Sort S in ascending order of x's.
We obtain $S = \{(x_{\sigma(1)}, y_{\sigma(1)}), (x_{\sigma(2)}, y_{\sigma(2)}), \dots, (x_{\sigma(m)}, y_{\sigma(m)})\}$ with $x_{\sigma(1)} \leq x_{\sigma(2)} \leq \dots \leq x_{\sigma(m)}$.
Generate a set Z containing values of x without repetition:
$Z = \{z_1, z_2, \dots, z_n\}$ with $z_1 = x_{\sigma(1)} < z_2 < \dots z_n = x_{\sigma(m)}$
Initialization step:
positive_left = 0; negative_left = 0;
positive_right = # points with label $y_i = 1$ with $i = \overline{1, n}$;
negative_right = # points with label $y_i = 0$ with $i = \overline{1, n}$;
if *negative_right == 0* **then** `// all samples are positive`
    $\theta^s_1 = z_1$; **return** $h_{\theta^s_1}$
if *positive_right == 0* **then** `// all samples are negative`
    $\theta^s_2 = z_1$; **return** $h_{\theta^s_2}$

We consider:

$Loss_1(\theta) = \frac{\#\ left\_positive_\theta + \#\ right\_negative_\theta}{n}$ = error of h_{θ_1} at threshold θ where $h_{\theta_1} \in \mathcal{H}_1$

$Loss_2(\theta) = \frac{\#\ left\_negative_\theta + \#\ right\_positive_\theta}{n}$ = error of h_{θ_2} at threshold θ where $h_{\theta_2} \in \mathcal{H}_2$

Compute initial $Loss_1$ and $Loss_2$ values

$\theta_1^s = z_1$; $\theta_2^s = z_1$; $L_1 = Loss_1(\theta_1^s)$; $L_2 = Loss_2(\theta_2^s)$

for $i \leftarrow 2$ **to** n **in**

```
    // updating positive and negative counts on both sides
    // for each change of threshold
    if y_i == 0 then
        negative_left = negative_left +1
        negative_right = negative_right -1
    if y_i == 1 then
        positive_left = positive_left +1
        positive_right = positive_right -1
    // recomputing Loss values for each H class for the new threshold
    L_1^i = Loss_1(z_i)
    L_2^i = Loss_2(z_i)
    // selecting threshold with the best error for the both H classes
    if L_1^i < L_1 then
        θ_1^s = z_i
        L_1 = L_1^i
    if L_2^i < L_2 then
        θ_2^s = z_i
        L_2 = L_2^i
```

if $L_1 < L_2$ **then**

```
    return h_{θ_1^s}
```

else

```
    return h_{θ_2^s}
```

Complexity:

1. sorting: $O(m \cdot log(m))$
2. computing Z: $O(m)$
3. initialization step: $O(m)$
4. computing loss = $O(1)$
5. selecting best threshold: $O(m)$

Total Complexity: $O(m \cdot log(m))$

Point (c): We know that $\tau_{\mathcal{H}}(m) = \max_{C \subseteq X : |C| = m} |\mathcal{H}_C|$ and $\mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2$

$\implies \tau_{\mathcal{H}}(m) = \tau_{\mathcal{H}_1 \cup \mathcal{H}_2}(m) = \max_{C \subseteq X : |C| = m} |\mathcal{H}_{1C}| + |\mathcal{H}_{2C}| - |\mathcal{H}_{1C} \cap \mathcal{H}_{2C}|$.

We consider $C = \{c_1, c_2, \dots, c_m\}$ a set of m points with $c_i \leq c_j$.

For $\mathcal{H}_{1C}$ we can have at most $m + 1$ different functions:

$a_1 < c_1 < a_2 < c_2 < \dots < a_m < c_m < a_{m+1} \implies |\mathcal{H}_{1C}| = \left|\{h_{a_1}, h_{a_2}, \dots h_{a_m}, h_{a_{m+1}}\}\right| = m + 1$. [1]

h_{a_1} labels points $c_1, c_2, \dots, c_m$ with labels $(1, 1, \dots, 1, 1)$.

h_{a_2} labels points $c_1, c_2, \dots, c_m$ with labels $(0, 1, \dots, 1, 1)$.

$\vdots$

h_{a_m} labels points $c_1, c_2, \dots, c_m$ with labels $(0, 0, \dots, 0, 1)$.

$h_{a_{m+1}}$ labels points $c_1, c_2, \dots, c_m$ with labels $(0, 0, \dots, 0, 0)$.

For $\mathcal{H}_{2C}$ we have at most $m + 1$ different functions:

$b_1 < c_1 < b_2 < c_2 < \dots < b_m < c_m < b_{m+1} \implies |\mathcal{H}_{2C}| = \left|\{h_{b_1}, h_{b_2}, \dots h_{b_m}, h_{b_{m+1}}\}\right| = m + 1$. [2]

h_{b_1} labels points $c_1, c_2, \dots, c_m$ with labels $(0, 0, \dots, 0, 0)$.

h_{b_2} labels points $c_1, c_2, \dots, c_m$ with labels $(1, 0, \dots, 0, 0)$.

$\vdots$

h_{b_m} labels points $c_1, c_2, \dots, c_m$ with labels $(1, 1, \dots, 1, 0)$.

$h_{b_{m+1}}$ labels points $c_1, c_2, \dots, c_m$ with labels $(1, 1, \dots, 1, 1)$.

We can observe that h_{a_1} is equivalent to $h_{b_{m+1}}$ and $h_{a_{m+1}}$ is equivalent to h_{b_1}. [3]

From [1], [2] and [3] we have $\tau_{\mathcal{H}_1 \cup \mathcal{H}_2}(m) = m + 1 + m + 1 - 2 = 2m$.

Point (d): ***Lemma Sauer-Shelah-Perles***

Let $\mathcal{H}$ be a hypothesis class with $VC\dim(\mathcal{H}) \leq d < \infty$. Then, for all m, we know that:

$$\tau_{\mathcal{H}}(m) \leq \sum_{i=0}^{d} C_m^i$$

In particular, if $m > d + 1$ then $\tau_{\mathcal{H}}(m) \leq (\frac{em}{d})^d = O(m^d)$

We know that VC $\dim(\mathcal{H}_{\text{thresholds}}) = 1$, based on that fact it's trivial to prove that $VC\dim(\mathcal{H}) = VC\dim(\mathcal{H}_1 \cup \mathcal{H}_2) = 2$.

At the previous point, we proved that $\tau_{\mathcal{H}}(m) = 2m$ [1]

Considering $VCdim(\mathcal{H}) = 2 = d$ we have the following cases:

Case 1: $m > 3$, we have that $\tau_{\mathcal{H}}(m) \leq (\frac{em}{d}^d) \implies \tau_{\mathcal{H}}(m) \leq (\frac{em}{d}^2) \overset{[1]}{\implies} 2m \leq \frac{e^2 m^2}{4} \iff$ $8m \leq e^2 m^2 \mid \cdot \frac{1}{m}$ (because we know that m > 0) $\iff 8 \leq e^2 m \iff \frac{8}{e^2} \leq m \iff 1.082\dots \leq m$ (this holds for every m > 3) [2].

Case 2: $m = 1 \implies \tau_{\mathcal{H}}(m) \leq \sum_{i=0}^{2} C_m^i \implies \tau_{\mathcal{H}}(m) \leq C_m^0 + C_m^1 + C_m^2 \implies \tau_{\mathcal{H}}(m) \leq 1 + m + \frac{m(m-1)}{2} = 1 + 1 + 0 = 2 \overset{[1]}{\implies} 2 \cdot m = 2 \cdot 1 = 2 \leq 2$ [3].

Case 3: $m = 2 \implies \tau_{\mathcal{H}}(m) \leq \sum_{i=0}^{2} C_m^i \implies \tau_{\mathcal{H}}(m) \leq C_m^0 + C_m^1 + C_m^2 \implies \tau_{\mathcal{H}}(m) \leq 1 + m + \frac{m(m-1)}{2} = 1 + 2 + 1 = 4 \overset{[1]}{\implies} 2 \cdot m = 2 \cdot 2 = 4 \leq 4$ [4].

Case 4: $m = 3 \implies \tau_{\mathcal{H}}(m) \leq \sum_{i=0}^{2} C_m^i \implies \tau_{\mathcal{H}}(m) \leq C_m^0 + C_m^1 + C_m^2 \implies \tau_{\mathcal{H}}(m) \leq 1 + m + \frac{m(m-1)}{2} = 1 + 3 + 3 = 7 \overset{[1]}{\implies} 2 \cdot m = 2 \cdot 3 = 6 \leq 7$ [5].

From [2], [3], [4], [5] we obtain that $\tau_{\mathcal{H}}(m) \leq \sum_{i=0}^{2} C_m^i$ as expected from Lemma Sauer-Shelah-Perles, to a more general form we have that $2m$ (our result obtained earlier) $\leq C_m^0 + C_m^1 + C_m^2 = 1 + m + \frac{m(m-1)}{2} = \frac{m^2+m+2}{2} \iff 2m \leq \frac{m^2+m+2}{2}$ for $m > 0$, which is correct, but is a high limit from the one computed.

Point (e): We know that $\tau_{\mathcal{H}}(m) = \underset{C \subseteq X : |C| = m}{max} |\mathcal{H}_C|$ and from the *Lemma Sauer-Shelah-Perles* we know that $\tau_{\mathcal{H}}(m) \leq \sum_{i=0}^{d} C_m^i$. Let's consider the definition from the lecture of $\mathcal{H}_{\text{thresholds}}$ such that

$$\mathcal{H}_{\text{thresholds}} = \left\{ h_a : \mathbb{R} \longrightarrow \{0,1\} \mid h_a(x) = \begin{cases} 1, & \text{x < a} \\ 0, & \text{otherwise} \end{cases} \right\}$$

We know that $VC\dim(\mathcal{H}_{\text{thresholds}}) = 1$, based on that fact we get:

$\tau_{\mathcal{H}_{\text{thresholds}}}(m) \leq \sum_{i=0}^{1} C_m^i = C_m^0 + C_m^1 = 1 + m$ [1]

Let's consider $C = \{c_1, c_2, \ldots, c_m\}$ a set of m points, with $c_i < c_j$.

If we take $a_1 < c_1 < a_2 < c_2 < \ldots < c_m < a_{m+1}$ then $\mathcal{H}_C$ will have $|\mathcal{H}_C| = \left|\{h_{a_1}, h_{a_2}, \ldots, h_{a_{m+1}}\}\right| = m + 1$ (there will be at most $m + 1$ functions, from *Lecture 8*). [2]

From [1] and [2] we conclude that $\mathcal{H}_{\text{thresholds}}$ has the growth function ($\tau_{\mathcal{H}}(m)$) equal to the general upper bound.

Exercise 2

Statement:

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 $\mathbf{D}^{(1)}$, weak classifier h_1 with error ε_1, at round 2 we obtain distribution $\mathbf{D}^{(2)}$, weak classifier h_2 with error ε_2).
- in the third round we compute for each i = 1, 2, …, m:

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

  where Z is a normalization factor such that $\mathbf{D}^{(3)}$ 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 $\varepsilon_t \leq 1/2 - \gamma_t$, $\gamma_t > 0$.

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

Solution:

Point (a): We know that $\varepsilon_t \leq \frac{1}{2} - \gamma,\ \gamma > 0$ for $t = 1, 2, 3$
For $t = 1$ let's assume that $\varepsilon_t = 0$ (our estimator does not make any mistakes, x_i is correctly classified, with $i = \overline{1, m}$) $\implies \underset{i \sim D^{(t)}}{Pr}[h_t(x_i) \neq y_i] = \sum_{i=1}^{m} D^{(t)} \cdot \mathbf{1}_{[h_t(x_i) \neq y_i]} = 0$

Based on that, we have the distribution at iteration $t + 1$:

$$D^{(t+1)}(i) = \frac{D^{(t)}(i) \cdot e^{-w_t}}{Z_{t+1}} = \frac{D^{(t)}(i) \cdot \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}}}{Z_{t+1}} \quad [1]$$

where $Z_{t+1} = \sum_{i=1}^{m} D^{(t)}(i) \cdot e^{-w_t h_t(x_i) y_i} = \sum_{i=1}^{m} D^{(t)}(i) \cdot e^{-w} = \sum_{i=1}^{m} D^{(t)}(i) \cdot \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} = \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} \cdot \sum_{i=1}^{m} D^{(t)}(i) \xrightarrow[distribution]{D^{(t)} probability}$
$\sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} \cdot 1 = \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}}$ [2].

From [1] and [2] we have $D^{(t+1)}(i) = \frac{D^{(t)}(i) \cdot \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}}}{\sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}}} = D^{(t)}(i)$, for i = $\overline{1, m}$.

This implies that if the first estimator has $\varepsilon_t = 0 \implies D^{(t+1)} = D^{(t)} \implies$ the estimator chosen at $t + 1$ will be the same as the one at iteration t (the sample importance has not change)
$\implies Pr[h_t = h_{t+1} \mid \varepsilon_t = 0] = 1$ [3]

For the general case when we have $\varepsilon_t \in (0, \frac{1}{2})$ we consider $D^{(t)}$ the probability distribution of importance to contain correctly classified and misclassified examples.

We can write this in a more formal manner as:

$$\sum_{i=1}^{m} D^{(t)}(i) \cdot e^{-w_t h_t(x_i) y_i} = \sum_{i=1}^{m} D^{(t)}(i) \cdot e^{-w_t} \cdot \mathbf{1}_{[h_t(x_i)=y_i]} + \sum_{i=1}^{m} D^{(t)}(i) \cdot e^{w_t} \cdot \mathbf{1}_{[h_t(x_i)\neq y_i]}$$

We want to prove that $D^{(t)} \neq D^{(t+1)}$ because from that we have $h_t \neq h_{t+1}$.

For the misclassified samples we have that: $\sum_{i=1}^{m} D^{(t)}(i) \cdot e^{-w_t h_t(x_i) y_i} \cdot \mathbf{1}_{[h_t(x_i)\neq y_i]} = \sum_{i=1}^{m} D^{(t)}(i) \cdot e^{w_t} \cdot \mathbf{1}_{[}h_t(x_i) \neq y_i] = \sum_{i=1}^{m} D^{(t)}(i) \cdot \mathbf{1}_{[}h_t(x_i) \neq y_i] \cdot e^{w_t} = \varepsilon_t \cdot e^{w_t} = \varepsilon_t \cdot \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} = \sqrt{\varepsilon_t(1-\varepsilon_t)}$ [4]

In similar manner we have the normalization factor:

$$\sum_{i=1}^{m} D^{(t)}(i) \cdot e^{-w_t h_t(x_i) y_i} = \varepsilon_t \cdot \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} + (1-\varepsilon_t) \cdot \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} = 2\sqrt{\varepsilon_t(1-\varepsilon_t)} \text{ [5]}$$

By dividing [4] to [5] we get:

$$\sum_{i=1}^{m} D^{(t+1)}(i) \cdot \mathbf{1}_{[}y_i \neq h_t(x_i)] = \frac{\sqrt{\varepsilon_t(1-\varepsilon_t)}}{2\sqrt{\varepsilon_t(1-\varepsilon_t)}} = \frac{1}{2}$$

We can observe how in $D^{(t+1)}$ the misclassified samples by h_t represent half of importance from the probability distribution $\Longrightarrow$ if $\varepsilon_t \in (0, \frac{1}{2}) \Longrightarrow D^{(t)} \neq D^{(t+1)}$ (because the misclassified samples will weight more in the next iteration) $\Longrightarrow h^{(t)} \neq h^{(t+1)}$
$\Longrightarrow Pr[h^{(t)} = h^{(t+1)} | \varepsilon_t \in (0, \frac{1}{2})] = 0$ [6]

From [3] and [6] $\Longrightarrow Pr[h^{(t)} = h^{(t+1)}] = \begin{cases} 1, & \varepsilon_t = 0 \\ 0, & \text{otherwise} \end{cases}$

Point (b): Assuming that the training error of h_{final} is at most $\frac{1}{2} - \frac{3}{2}\gamma + 2\gamma^2$ we will show that is strictly smaller than $\frac{1}{2} - \gamma$.

We have that $\frac{1}{2} - \frac{3}{2}\gamma + 2\gamma^3 \leq \frac{1}{2} - \gamma \iff \frac{3}{2}\gamma - 2\gamma^3 \geq \gamma \iff \frac{1}{2}\gamma - 2\gamma^3 \geq 0 \iff$
$\frac{1}{2}\gamma(1 - 4\gamma^2) \geq 0 \iff \frac{1}{2}\gamma(1-2\gamma)(1+2\gamma) \geq 0$ [1].

Based on the fact we know $\varepsilon_t \in (0, \frac{1}{2}) \Longrightarrow \gamma \in (0, \frac{1}{2})$ [2].

From [1] and [2] we conclude that
$\frac{1}{2}\gamma > 0,\ 1 - 2\gamma > 0,\ 1 + 2\gamma > 0 \Longrightarrow \frac{1}{2}\gamma \cdot (1-2\gamma)(1+2\gamma) \geq 0 \Longrightarrow \frac{1}{2} - \frac{3}{2}\gamma + 2\gamma^2 \leq \frac{1}{2} - \gamma$

Exercise 3

Statement:

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

a. Give an upper bound (any upper bound that you can come up) of the VCdimension of $\mathcal{H}$ in terms of $|\Sigma|$ 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?

Example: let Σ = {a, b, c}, m = 4 and the training set S = {(aabc, 1), (baca, 0), (bcac,0), (abba, 1)}. The output of the algorithm should be h_{ab}.

Solution:

Point (a): From *Lecture 6* we know that $VC\dim(\mathcal{H}) \leq \log_2 |\mathcal{H}|$.

Let's consider $\Sigma = \{a, b\}$ this implies that the sample space

$$\chi = \{a,b\}^m = \{\underbrace{aaa\ldots a}_{\text{m}},\ \underbrace{baa\ldots a}_{\text{m}},\ \underbrace{bba\ldots a}_{\text{m}},\ \ldots,\underbrace{bbb\ldots b}_{\text{m}}\} \text{ with } |\chi| = 2^m$$

$\mathcal{H} = \{h_w : \Sigma^m \longrightarrow \{0,1\},\ w \in \Sigma^*,\ 0 < |w| \leq m,$ s.t $h_h(x) = 1$ if w is a substring of x$\}$
$\iff \mathcal{H} = \{h_w : \{a,b\}^m \longrightarrow \{0,1\},\ w \in \{a,b\}^*,\ 0 < |w| \leq m,$ s.t $h_h(x) = 1$ if w is a substring of x$\}$

$$\iff \mathcal{H} = \left\{ \begin{array}{l} h_w : \{a,b\}^m \longrightarrow \{0,1\} \mid w \in \{a, b, aa, ab, ba, bb, \ldots, \underbrace{bb\ldots b}_{\text{m times}}\} \\ \\ h_w(x) = 1, \text{ if w is substring of x} \end{array} \right\}$$

We have that $|\mathcal{H}| = \left|\{h_a, h_b, h_{aa}, h_{ab}, h_{ba}, h_{bb}, \ldots, h_{bb\ldots b}\}\right| = 2 + 4 + 8 + \ldots + 2^m = \sum_{i=1}^{m} 2^i = 2 \cdot \frac{2^m - 1}{2-1} = 2^{m+1} - 2$.

To generalize, if $\left|\Sigma\right| = k \in \mathbb{N}^*,\ \Sigma = \{c_i \mid i \in \overline{1,k}\} \Longrightarrow \chi = \{c_i \mid i \in \overline{1,k}\}^m \Longrightarrow |\chi| = k^m = |\Sigma|^m$.

$|\mathcal{H}| = \left|\{h_{c_1}, h_{c_2}, \ldots h_{c_k}, h_{c_1c_1}, h_{c_1c_2}, \ldots, h_{c_kc_k}, \ldots \underbrace{h_{c_kc_k\ldots c_k}}_{\text{m times}}\}\right| \Longrightarrow k^1 + k^2 + k^3 + \ldots k^m = \sum_{i=1}^{m} k^i = k \cdot \frac{k^m - 1}{k-1} = \frac{k^{m+1} - k}{k-1}$, where $k = |\Sigma|$

$\Longrightarrow |\mathcal{H}| = \frac{|\Sigma|^{m+1} - |\Sigma|}{|\Sigma| - 1}$

$\Longrightarrow VC\dim(\mathcal{H}) \leq \log_2 |\mathcal{H}| = \log_2 \frac{|\Sigma|^{m+1} - |\Sigma|}{|\Sigma| - 1}$

Point (b): We are under the realizable case $\Longrightarrow$
Given a training set $S = \{(x_1, y_1), (x_2, y_2), \dots (x_n, y_n) \mid x_i \in \Sigma^m, y_i \in \{0, 1\}, i = \overline{1, n}\}$ exists a function $h_w \in \mathcal{H}$ with $w \in \Sigma^*, 0 < |w| \leq m$ such that $y_i = h_{w^*}(x_i)$.
Let's consider the following algorithm for our hypothesis class $\mathcal{H}$:

Input: S
Output: h_{w^s}

positive_words = [word for (word,y) in S if y == 1]
negative_words = [word for (word,y) in S if y == 0]

P = length(positive_words)

if P == *0* **then**
    w_s = word $\notin$ negative_words, with | word | = m
    return h_{w_s}

positive_substrings = dict() `// initialize empty dict`
negative_substrings = set() `// initialize empty set`

`// We need to find the substrings that appear in all`
`// p positive words`
for *word* ***in*** *positive_words* **do**
    substrings = generate_all_substrings(word)
    for *substring* ***in*** *substrings* **do**
        positive_substrings[substring] += 1
    end
end

`// Next we need to generate all substrings from negative words`
`// Such that the selected W it's not contained in them`
for *word* ***in*** *negative_words* **do**
    substrings = generate_all_substrings(word)
    for *substring* ***in*** *substrings* **do**
        negative_substrings.update(substring)
    end
end
`// Finally, we select W as the first substring that appears in all`
`// positive examples and it's not in the set of negative substrings`
for *substring, counts* ***in*** *positive_substrings.items()* **do**
    if *substring* ***not in*** *negative_substrings* ***and*** *count* == P **then**
        return $h_{substring}$
end

Complexity:

1. Initialization step: $O(n)$
2. Iteration through positive and negative words: $O(n)$
3. Generationg all substrings for a word of size m: $O(m^2)$
4. Finding the W_S: $O(n^2 \cdot m^4)$
    - iterating through $n \cdot m^2$ substrings: $O(n \cdot m^2)$
    - searching substring in maximum $n \cdot m^2$ possible negative substrings: $O(n \cdot m^2)$

Total Complexity: $O(n^2 \cdot m^4)$