Assignment 2

Deadline: Year 1 - Sunday, 19^{th} of June, 23:59,
Student: Lucian Istrati - Group 411 - Data Science

Upload your solutions as a zip archive at:
https://tinyurl.com/AML-2022-ASSIGNMENT2

1. **(1.5 points)** Consider $\mathcal{H}$ the class of 3-piece classifiers (signed intervals):

$$\mathcal{H} = \{h_{a,b,s} : \mathbb{R} \to \{-1, 1\} \mid a \leq b, s \in \{-1, 1\}\}, \text{ where } h_{a,b,s}(x) = \begin{cases} s, & x \in [a, b] \\ -s, & x \notin [a, b] \end{cases}$$

   a. Compute the shattering coefficient $\tau_H(m)$ of the growth function for $m \geq 0$ for hypothesis class $\mathcal{H}$. **(1 point)**

   b. Compare your result with the general upper bound for the growth functions and show that $\tau_H(m)$ obtained at previous point a is not equal with the upper bound. **(0.25 points)**

   c. Does there exist a hypothesis class $\mathcal{H}$ for which is equal to the general upper bound (over or another domain $\mathcal{X}$)? If your answer is yes please provide an example, if your answer is no please provide a justification. **(0.25 points)**

Solution 1. a) In the 3rd Seminar exercise number 4 it was shown that VCdim(H) is equal to 3. Based on this reference "https://en.wikipedia.org/wiki/Growth_function" (the Sauer-Shelah lemma) we know that if VCdim(H) equals to d, then for any m Growth(H, m) <=

$$\sum_{i=0}^{d} \binom{m}{i}$$

, the upper bound for finite VCdimension. In our case since VCdim(H) = 3, we particularise: Growth(H, m) <=

$$\sum_{i=0}^{3} \binom{m}{i}$$

- which is the general upper bound in our case. By extending the calculus of

$$\sum_{i=0}^{3} \binom{m}{i}$$

we obtain that this expression is equal to: $1 + 5/6 * m + 1/6 * m^3$, so essentially our upper bound is polynomial with the degree 3.

By taking a couple of possible values for m and trying to see what they can label we obtain the following examples:

- m = 1
- -1
- 1,
- m = 2
- -1, 1,
- 1, -1
- -1, -1
- 1, 1
- m = 3
- -1, -1, 1,
- -1, 1, -1
- -1, -1, -1
- -1, 1, 1
- 1, -1, 1,
- 1, 1, -1
- 1, -1, -1
- 1, 1, 1
- m = 4

- -1, -1, -1, 1,
- -1, -1, 1, -1
- -1, -1, -1, -1
- -1, -1, 1, 1
- -1, 1, -1, 1 $\rightarrow$ impossible, cannot be shattered
- -1, 1, 1, -1
- -1, 1, -1, -1
- -1, 1, 1, 1
- 1, -1, -1, 1,
- 1, -1, 1, -1 $\rightarrow$ impossible, cannot be shattered
- 1, -1, -1, -1
- 1, -1, 1, 1
- 1, 1, -1, 1,
- 1, 1, 1, -1
- 1, 1, -1, -1
- 1, 1, 1, 1

So, from these examples we observe that any combination of -1s and 1s can be shattered except for the alternate combinations where the alternation occurs more than twice, while the rest types of labels such as: (-1); (1); (-1, 1, -1); (1, -1, 1); (-1, -1, 1, 1); (-1, 1, 1, -1); (1, -1, -1, 1) etc. are permitted because they can be shatters by hypothesis belonging to H class.

Now, let us calculate a set of values for a couple of m values:

- $m = 1 \rightarrow 2$
- $m = 2 \rightarrow 4$
- $m = 3 \rightarrow 8$
- $m = 4 \rightarrow 14$

Since the upper bound is a function of 3rd degree, we conclude that we also need a function of 3rd degree in this case, we obtain the following system:→

$c_1 + c_2 + c_3 + c_4 = 2$

$8 * c_1 + 4 * c_2 + 2 * c_3 + c_4 = 4$

$27 * c_1 + 9 * c_2 + 3 * c_3 + c_4 = 8$

$64 * c_1 + 16 * c_2 + 4 * c_3 + c_4 = 14$

After solving this system with an online equation solver we obtain the following result: $c_1 = 0, c_2 = 1, c_3 = -1, c_4 = 2$

So the actual $\tau_{H(m)}$ is $m^2 - m + 2$

To conclude, we observe from the second set of examples that the shattering coefficient of $\tau_H(m)$ is the maximal number of different functions from a set C with the size m to -1 and 1 possible labels that can be obtained by limiting H to C and can be bounded by ... and $\tau_H(m)$ is equal to $m^2 - m + 2$.

Solution 1. b) The general upper bound is $1+5/6*m+1/6*m^3$. Since $\tau_H(m)$ is equal to $m^2 - m + 2$ so essentially what we are supposed to do is to prove that $m^2 - m + 2$ is strictly smaller than $1 + 5/6 * m + 1/6 * m^3$ and cannot be equal. This might be equal or the first might be smaller than the latter in a few cases (explicitly tackling the case where $m >= 0$ because is the only one we are interested in), but for any m between 1 and 2 and higher than 3, it holds, by constructing the function of differences we obtain $1/6 * m^3 - m^2 + 11/6 * m - 1$. Solving the equation of this function we obtain the roots 1, 2 and 3 and the function is negative only when m is smaller than 1 or when m is between 2 and 3, in the rest of the cases being positive. Since we are discussing about the general case, we will simply calculate the limit of the function $1/6*m^3-m^2+11/6*m-1$ when m goes to +infinity and we will obtain +infinity, thus the strict inequality holds for the case in which we are interested and the growth function obtained at point a) is not equal to the general upper bound.

Solution 1. c)$1+5/6*m+1/6*m^3$ In order to check whether a hypothesis class H for which $\tau_H(m)$ is equal to the general upper bound we will pass a couple of values through this function in order to test what a possible hypothesis class could satisfy this function:

- $1 \rightarrow 2.0 = 2^1$
- $2 \rightarrow 4.0 = 2^2$
- $3 \rightarrow 8.0 = 2^3$

- $4 \rightarrow 15.0 = 2^4 - 1$
- $5 \rightarrow 26.0 = 2^5 - 6$

Intuitively we can see that up to 3 all possible mappings to -1, 1 are satisfied, so then we must choose a function that definitely satisfies all the posibilities, a good start would be another interval function like h(a,b,s). However, the difference starts to arise for m = 4 and m = 5, the difference is though that for m = 4 we have a 16 - 1 result instead of a 16 - 2 result and for m = 5 we have a 32 - 6 result instead of a 32 - 10 result, so then we must conclude that this new interval function should support a higher complexity than the one h(a,b,s) has. First step would be to add a new parameter and thus accomodating new types of labellings. First idea might be something like this:

$$\mathcal{H} = \{h_{a,b,c,s} : \mathbb{R} \rightarrow \{-1, 1\} \mid a \leq b, c, s \in \{-1, 1\}\}, \text{ where } h_{a,b,c,s}(x) = \begin{cases} s, & x \in [a, b) \\ s, & x \in [c, +inf) \\ -s, & x \in (-inf, a) \\ -s, & x \in [b, c) \end{cases}$$

But, for m = 4 this would effectively satisfy all the possible 16 mappings, since it would satisfy both (-1, 1, -1, 1) and (1, -1, 1, -1). So then, a limitation is needed, the most natural one would be to restrict either of the 2 above mentioned possibilities(in order to have 15 mappings), let's say that we choose s to be just 1, then we would accomodate (1,-1,1,-1), but not (-1, 1, -1, 1).

So, to answer the question, yes, there is another hypothesis class H for which $\tau_H(m)$ is equal to the general upper bound and the example for this is the following:

$$\mathcal{H} = \{h_{a,b,c,s} : \mathbb{R} \rightarrow \{-1, 1\} \mid a \leq b, c, s \in \{1\}\}, \text{ where } h_{a,b,c,s}(x) = \begin{cases} s, & x \in [a, b) \\ s, & x \in [c, +inf) \\ -s, & x \in (-inf, a) \\ -s, & x \in [b, c) \end{cases}$$

2. **(1.5 points)** Consider the concept class C_2 formed by the union of two closed intervals $[a, b] \cup [c, d]$, where $a, b, c, d \in \mathbb{R}, a \leq b \leq c \leq d$. Give an efficient ERM algorithm for learning the concept class C_2 and compute its complexity for each of the following cases:

a. realizable case. **(1 point)**

b. agnostic case. **(0.5 point)**

Solution 2. a)

First, let's treat the realizable case scenario, for this, there should be a function h(a', b', c', d', x) = (1[a', b'] U 1[c', d'])(x) which will be the labelling functions for the points belonging to the training set:

S = (x1, y1), (x2, y2), ..., (xm, ym) where y_i = h(a', b', c', d', xi).

Then, we will sort the points, based on x's (we will then consider a permutation $\sigma(i)$ with i from 1 to m where this permutation is the permutation of the sorted indexes):

S = (xσ(1), yσ(1)), (xσ(2), yσ(2)), ..., (x$\sigma(m)$, y$\sigma(m)$)

with:

$x_\sigma(1) <= x_\sigma(2) <= x_\sigma(m)$

Having done this, we will define the steps for the ERM algorithm in order to discover a, b, c, d after the sorting of S.

0. We sort the training set S based on x's → with a complexity of O(m * log(m)) assuming we use an efficient sorting algorithm

1. First, we will look to check if we have both negative and positive labels

1. We have just positive labels, then we will stop and return h(a, b, c, d) with $a = x_\sigma(1)$ and $b = c = d = x_\sigma(m)$ → This step will have a complexity of O(m)

2. We have just non-positive labels, then we will also stop and return h(a, b, c, d) with $a = b = x_\sigma(1)$ - ϵ and $c = d = x_\sigma(m) + \epsilon$, where we define ϵ as a strictly positive number higher than 0 → This step will have a complexity of O(m)

3. Otherwise we have both positive and non-positive labels, then we must fix a, b, c and d. a and b will be set as the smallest x_i for which $y_i = +1$ while c and d will be set as the highest y_i for which $y_i = +1$ → this step will have a complexity of O(2 * m) which is in the order of O(m)

4. Since we found a,b,c and d, but a=b and c=d we will then need to set b and c, since b can be higher than a and c can be smaller than d, thus they do not have to be equal. In order to find the value for b we will look for the highest $x_\sigma(i-1)$ for which $y_\sigma(i)$ = -1 and $y_\sigma(i-1) = +1$. In order to find the value for c we will look for the highest $x_\sigma(i)$ for which $y_\sigma(i) = +1$ and $y_\sigma(i-1)$ = -1. → Also a complexity of O(2*m) which is in the order of O(m)

5. Lastly we will return h(a, b, c, d) → $O(1)$

By adding up the complexities of which step in the algorithm we observer it is a linear combination of linear and linear times logarithmic functions, thus the product between linear and log being dominant the final complexity will be in O(m * log(m)).

Solution 2. b)

For the agnostic scenario, we will work with a distribution D over X x 0, 1, as we will have 2^m labelling functions for m points.

First step of the algorithm will be to also sort the S set based on the x values.

Then, after we are doing to sorting we will also need to define what the loss will be for our identified interval.

We define the Loss(a, b, c, d) as the sum between the number of negative points in [a,b] and [c,d] with the number of positive points which lay outside of the aforementioned intervals while dividing this at m - the size of S.

We will also need to precompute 2 matrices $pos_{between}$ and $neg_{between}$ where an element at the indexes (i, j) will be defined as the number of positive labels between the ith and jth x-value, while the latter being the number of negative labels betwene the ith and jth x-value.

By having these precomputed, with the indexes i,j,k and l $\rightarrow$ we will be able to tell that: $Loss(i, j, k, l) = (pos_{between}[1][i-1] + neg_{between}[i][j] + pos_{between}[j+1][k-1] + neg_{between}[k][l] + pos_{between}[l+1][m])/m$.

Having said this we define the following ERM algorithm:

0. We sort the training set S based on x's $\rightarrow$ with a complexity of O(m * log(m)) assuming we use an efficient sorting algorithm

1. We will check if we have positive and negative values. $\rightarrow$ O(m)

2. If we have just positive values, then we will stop and return h(a,b,c,d) with a = $x_\sigma(1)$ and b = c = d = $x_\sigma(m)$ $\rightarrow$ This step will have a complexity of O(m)

3. If we have just negative values, then we will also stop and return h(a,b,c,d) with a = b = $x_\sigma(1)$ - ϵ and c = d = $x_\sigma(m)$ + ϵ, where we define ϵ as a strictly positive number higher than 0 $\rightarrow$ This step will have a complexity of O(m),

4. Otherwise if we have positive and negative values and we continue on the third branch

5. We define $i_{final}, j_{final}, k_{final}, l_{final}, min_{loss} = 1$

6. We compute $pos_{between}$ as $pos_{between}[i][j]$ = number of 1's between i and $\rightarrow$ O(m^2)

7. We compute $neg_{between}$ as $neg_{between}[i][j]$ = number of -1's between i and j $\rightarrow$ O(m^2)

8.

- for i in 1, n
- for j in 1,n
- for k in i, n
- for l in 1, n
- Loss(i, j, k, l) = $(pos_{between}[1][i-1] + neg_{between}[i][j] + pos_{between}[j+1][k-1] + neg_{between}[k][l] + pos_{between}[l+1][m])/m$
- if Loss(i, j, k, l) $< min_{loss}$:
- min_{loss} = Loss(i, j, k, l)
- i_{final} = i
- j_{final} = j
- k_{final} = k
- l_{final} = l

$\rightarrow O(\text{m}^{\ 4})$ 9. Return h(i_{final}, j_{final}, k_{final}, l_{final}) $\rightarrow O(1)$

By adding up the complexities of which step in the algorithm we observer it is a linear combinatios of linear, quadratic and cuaternary complexities, with the latter being dominant the final complexity will be in O(m^4).

3. **(1j.5 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 $\mathbf{D}^{(1)}$, 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 $i = 1, 2, \ldots, m$:

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

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

a. What is the probability that the classifier h_1 (selected at round 1) will be selected again at round 2? Justify your answer. **(0.75 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 $\frac{1}{2} - \frac{3}{2}\gamma + \gamma^2$ and show that this is strictly smaller than $\frac{1}{2} - \gamma$. **(0.75 points)**

Solution 3. a)

We should show that that if $\epsilon_i <= \frac{1}{2} - \gamma_i, \gamma_i > 0, i \in 1, 2, 3$ we will have a probability of 0 if choosing h_1 in the second round.

- $D_{t+1}(i) = \frac{D_t(i) * e^{(-w_t) * h_t(x_i) * y_i}}{Z_(t+1)}$
- Z_{t+1} is a normalization factor
- $w_t = \frac{1}{2} * log(\frac{1}{\epsilon_t} - 1)$
- $\epsilon_t = P_{i \sim D_t[h_t(x_i) \neq y_i]} = \sum_{h_t(x_i) \neq D_t(i)}$
- if $h_t(x_i) = y_i \rightarrow D_{t+1}(i) = \frac{(D_t(i) * e^{(-w_t)})}{Z_(t+1)} = \frac{D_t(i) * e^{\frac{-1}{2} * \ln(\frac{1}{\epsilon_t} - 1)}}{Z_{t+1}} = D_t(i) * \frac{\sqrt{\frac{\epsilon_t}{1-\epsilon_t}}}{Z_{t+1}}$
- if $h_t(x_i) \neq y_i \rightarrow D_{t+1}(i) = \frac{D_t(i) * e^{w_t}}{Z_{t+1}} = \frac{D_t(i) * e^{\frac{1}{2} * \ln(\frac{1}{\epsilon_t} - 1)}}{Z_{t+1}} = \frac{D_t(i) * (\sqrt{\frac{1-\epsilon_t}{\epsilon_t}})}{Z_{t+1}}$
- $Z_{t+1} = \sum_{h_t(x_i) = y_i} D_t(i)(\sqrt{\frac{\epsilon_t}{(1-\epsilon_t)}}) + \sum_{h_t(x_i) \neq y_i} D_t(i) * (\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}) = (1 - \epsilon_t) * (\sqrt{\frac{\epsilon_t}{(1-\epsilon_t)}}) + \epsilon_t * (\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}) = 2 * \sqrt{\epsilon_t * (1 - \epsilon_t)}$

If we assume that $h_2 = h_1$:

Then we have that $\epsilon_2 = \sum_{h_2(x_i) \neq y_i} D_2(i) = \sum_{h_1(x_i) \neq y_i} D_2(i) = \sum_{h_1(x_i) \neq y_i} \frac{(D_1(i) * \sqrt{\frac{1-\epsilon_1}{\epsilon_1}})}{Z_2} = \sum_{h_1(x_i) \neq y_i} \frac{D_1(i) * \sqrt{\frac{(1-\epsilon_1)}{\epsilon_1}}}{2 * \sqrt{\epsilon_1 * (1-\epsilon_1)}} = \frac{(\sqrt{\frac{(1-\epsilon_1)}{\epsilon_1}} * \sum_{h_1(x_i) \neq y_i} D_1(i))}{2 * \sqrt{\epsilon_1 * (1-\epsilon_1)}} = \frac{\sqrt{\frac{1-\epsilon_1}{\epsilon_1}} * \epsilon_1}{2 * \sqrt{\epsilon_1 * (1-\epsilon_1)}} = \frac{\sqrt{\frac{1-\epsilon_1}{\epsilon_1}} * \epsilon_1 * \sqrt{\epsilon_1 * (1-\epsilon_1)}}{2 * \epsilon_1 * (1-\epsilon_1)} = \frac{(1-\epsilon_1)}{2 * (1-\epsilon_1)} = \frac{1}{2}$. And from the hypothesis it is known

ϵ_2 is smaller than or equal to $\frac{1}{2} - \gamma_2$, but we obtained that $\epsilon_2 = \frac{1}{2}$ which results in a contradiction. It is a contradiction because we know that $\gamma_i > 0$, for $i \in 1, 2, 3$ from the hypothesis. Therefore, since we obtained this contradiction, the event cannot happen and it is clear that the probability to select the classifier h_1 at round 2 again is 0.

Solution 3. b) We will break these problems into its two tasks:

Solution 3.b).1)

We will break the problem by first showing that the training error of the final classifier h_{final} is at most $\frac{1}{2} - \frac{3}{2} * \gamma + \gamma^2$ where $\gamma = min\gamma_1, \gamma_2, \gamma_3$:

3.b).2)

Now that we know that h_{final} is at most $\frac{1}{2} - \frac{3}{2} * \gamma + \gamma^2$, we must now show the following: $\frac{1}{2} - \frac{3}{2} * \gamma + \gamma^2 < \frac{1}{2} - \gamma$ where $\gamma = min\gamma_1, \gamma_2, \gamma_3$.

By moving the terms on the left-hand-side we obtain the following function g, defined as: $g(\gamma) = \gamma^2 - \frac{1}{2} * \gamma$. After solving the equation of this function we obtain the following solutions for $g(\gamma) = 0$ with $\gamma_1 = 0$ and $\gamma_2 = \frac{1}{2}$ with function being negative when γ higher than 0 and γ is also smaller than 0.5 and positive otherwise(when γ is smaller than 0 or higher than 0.5). From the course, we know that in weak learnability we only need to output a hypothesis whose error rate is at most $\frac{1}{2}\gamma$, namely, whose error rate is slightly better than what a random labeling would give us, so this would make our γ smaller than 0.5 and simultaneously γ has to be higher than 0.0 from the hyptothesis, thus the value of the function $g(\gamma) = \gamma^2 - \frac{1}{2} * \gamma$ is negative and $\frac{1}{2} - \frac{3}{2} * \gamma + \gamma^2 < \frac{1}{2} - \gamma$ where $\gamma = min\gamma_1, \gamma_2, \gamma_3$.

4. **(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 stronger 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.

Solution 4.

From the definition of γ-weak-learner given in the Boosting lecture from the course, we know that A learning algorithm, A, is a γ-weak-learner for a class H if there exists a function $m_H : (0,1)ßN$ such that:

- for every $\delta > 0$ (confidence)
- for every labeling $f \in H, f : X \to -1, +1$ (realizability case)
- for every distribution D over X

when we run the learning algorithm A on a training set, consisting of $m >= m_H\delta$ examples sampled i.i.d. from D and labeled by f, the algorithm A returns a hypothesis h (h might not be from H - improper learning) such that, with probability at least $1 - \delta$ (over the choice of examples), $LD, f(h) <= \frac{1}{2} - \gamma$

A hypothesis class H is γ-weak-learner if there exists a γ-weak-learner for that class.

We will then use the distribution rule to obtain 2-term CNF formulae starting from a 2-term DNF formulae

$A_1 \vee A_2 = \wedge u \in A_1, v \in A_2(u \vee v) = \wedge y_{u,v} = \wedge(u \vee v)$.

By applying this distribution formula we will obtain a conjunction of $(2 * n)^2$ terms, with each term formed from a disjunction of 2 literals.

From the following lecture notes, lecture number 5 https://www.cs.princeton.edu/~mona/MachineLearning_lecture_notes.html we know that in order to learn give this week learner for 2-term DNF by a 2-term CNF we will try to showcase it with an Occam style algorithm which consists of the two following steps:

1. We will choose samples as using the formula for sample drawing: $m = poly(n, size(H))$
2. After picking up the samples we will try to discover a hypothesis which is consistent with the labeled examples, thus reducing the problem to learning inputs for our conjunctions. This will be done by introducing a set of labelled examples for $O(n^2)variables$, each one for each 2 terms CNF clause.

As mentioned in earlier the problem changed our problem from DNF to CNF and we need to learn conjunctions, thus we can solve this problem efficiently by letting the learning algorithm to output a hypothesis from the class $H_(conj)^(2 * n)^2)$.

Let's say that we perform this conjunction learning, then we consider these conjunctions over n boolean feature. There will be 3^n possibilities, since each feature can

receive a 0 as input, a 1 or cannot appear at all in a conjunction. Thus, $|H| = 3^n$, so a sufficient number of examples to learn a PAC concept is: $(\ln \frac{1}{\delta} + \ln 3^n)/\epsilon = (\ln \frac{1}{\delta} + n * \ln 3)/\epsilon$.

Consider conjunctions over n boolean features. There are 3 * n of these since each feature can appear positively, appear negatively, or not appear in a given conjunction. Therefore $|H| = 3 * n$.

Ex-oficio: 0.5 points.

Bibliography