CSCI-599 Machine Learning Theory
Assignment - 2

Sree Priyanka Uppu
(uppu@usc.edu)

1 PAC Learning Axis-aligned Rectangles in R

The objective is to learn an unknown axis-aligned rectangle R in an n-dimensional space. The player receives information about R only through the following process: a random point p is chosen according to some probability distribution D. The player is given the point p together with a label indicating whether p is contained in R (a positive example) or not contained in R (a negative example).

The goal of the player is to use as few examples as possible and as little computation as possible,to pick a hypothesis rectangle $R^{'}$ which is a close approximation to R. The error of $R^{'}$ is measured as the probability that a random point is chosen from D falls in the region $R \ \Delta \ R^{'}$, where $R \ \Delta \ R^{'}=$ (R - $R^{'}$) $\cup$ ($R^{'}$ - R).

We observe that the tightest fit rectangle $R^{'}$ is always contained in the target rectangle R. That is, $R^{'} \subseteq R$ and hence, $R \ \Delta \ R^{'}= R$ - $R^{'}$. For instance, the topmost of these strips which is shaded and denoted by $T^{'}$ in figure below is the region above the upper boundary of $R^{'}$ extended to the left and right, but below the upper boundary of R. Note that for an n dimensional rectangle there are 2n strips.Note that there is some overlap between the 2n rectangle strips at the corners. Now, if we can guarantee that the weight under D of each strip us at most ϵ/2n, then we can conclude that the error of $R^{'}$ is at most 2n(ϵ/2n)= ϵ.

Let us analyze the weight of the top strip $T^{'}$. Define T to be the rectangular strip along the inside top of R which encloses exactly weight ϵ/2n under D. Clearly, $T^{'}$ has weight exceeding ϵ/2n under D if and only if $T^{'}$ includes T. Furthermore, $T^{'}$ includes T if and only if no point in T appears in the sample S - since if S does contain as point $p \in T$, this point has a positive label since it is contained in R, and then by definition of the tightest fit, the hypothesis rectangle $R^{'}$ must extend upwards into T to cover p.

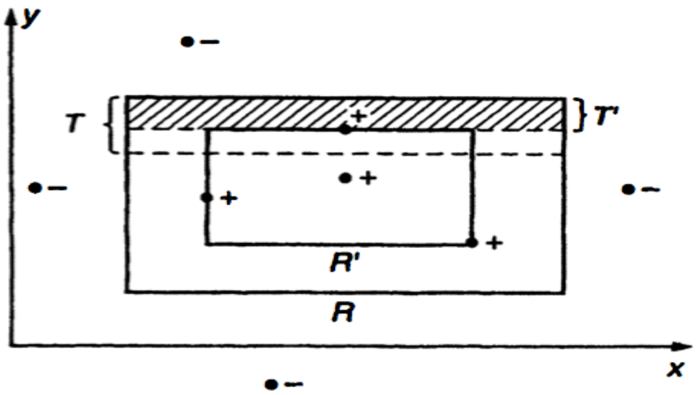

Figure 1: Analysis of the error contributed by the top strip $T^{'}$

By the definition of T, the probability that a single draw from the distribution D misses the region T is exactly 1-ϵ/2n. The probability that m independent draws from D all miss the region T is exactly $(1-\epsilon/2n)^m$ (Since, the probability of a conjunction of independent events is the product of the probabilities of the individual events). By Union bound, the probability that any of the 2n strips has weight greater than ϵ/2n is at most 2n$(1-\epsilon/2n)^m$.

Then,

$$2n(1-\epsilon/2n)^m \leq \delta$$

$$(1-\epsilon/2n)^m \leq \delta/2n$$

Taking natural log on both sides,

$$\ln(1-\epsilon/2n)^m \leq ln(\delta/2n)$$

$$m\epsilon/2n \geq -ln(\delta/2n)$$

From the inequality, -ln$(1-x) > x$ Thus,

$$m \geq (2n/\epsilon)ln(2n/\delta)$$

In summary, provided our tightest-fit algorithm takes a sample of at least $m \geq (2n/\epsilon)ln(2n/\delta)$ examples to form its hypothesis rectangle $R^{'}$, we can assert that with probability at least 1-δ, $R^{'}$ will mis classify a new point with the probability at most ϵ.

2 Two-oracle PAC model

a. Show that if H is PAC learnable then H is PAC learnable in the two-oracle model.

Assume that C is efficiently PAC Learnable using hypothesis class,H and a learning algorithm,L. Let h $\in$ H be the hypothesis output by the learner L and c $\in$ C be the target hypothesis. The distribution Dconsists of examples from both the D^+ and D^- examples. Hence,D=1/2(D^+ + D^-). Choose, δ such that:

$$Pr[error_D(h) \leq \epsilon/2] \geq 1 - \delta - (eq.1)$$

Evaluating $error_D(h)$,

$$error_D(h) = Pr_{x \sim D}[h(x) \neq c(x)]$$

$$error_D(h) = 1/2(Pr_{x \sim D^-}[h(x) \neq c(x)] + Pr_{x \sim D^+}[h(x) \neq c(x)])$$
$$error_D(h) = 1/2(error_{D^-}(h) + error_{D^+}(h))$$

Replace the value of $error_D(h)$ in eq.1, we get

$$Pr[error_{D^-}(h) \leq \epsilon/2] \geq 1 - \delta \quad and \quad Pr[error_{D^+}(h) \leq \epsilon/2] \geq 1 - \delta$$

which implies that both of them are PAC learnable as their sample complexity if polynomial in (ϵ, δ). Hence, H is PAC learnable in 2 oracle model with the sample complexity.

b. Show that if H is PAC learnable in the two-oracle model, then H is PAC learnable in the standard one-oracle model.

H is PAC learnable in two-oracle model, which implies that there exists a learning algorithm A such that for c$\in$C, ϵ>0 and δ>0 there exists m_- and m_+ polynomial in 1/ϵ,1/δ and size (c) such that if we dram m_- negative examples or more and m_+ positive examples or more, with confidence 1-δ, the hypothesis h output by A verifies:

$$Pr[error_{D^-}(h)] \leq \epsilon] \quad and \quad Pr[error_{D^+}(h)] \leq \epsilon]$$

Let D be the probability distribution over negative and positive examples. If we could dram m examples such that m $\geq$ max{m_-,m_+} and m is polynomial in $1/\epsilon$,$1/\delta$ and size (c) then the 2 Oracle PAC learning implies the standard oracle PAC learning model.

$$Pr[error_D(h)] \leq Pr[error_D(h)|c(x) = 0]Pr[c(x) = 0]+Pr[error_D(h)|c(x) = 1]Pr[c(x) = 1]$$

$$\leq \epsilon(Pr[c(x) = 0] + Pr[c(x) = 0]) \leq \epsilon$$

There are 2 possible cases with respect to the distribution:
Case (a.) The distribution D is not too biased that is the probability of drawing a positive examples is more than ϵ or the probability of drawing a negative examples is more than ϵ.

Let us determine the probability that we do not have enough of positive examples. Let S_m be the number of positive examples obtained when drawing m examples and the probability of drawing a positive example is ϵ Consider the probability that we draw fewer than m positive examples in q calls to EX. This probability can be bounded by the Chernoff bounds:

$$Pr[S_m \leq (1 - \alpha)m\epsilon] \leq e^{-m\epsilon\alpha^2}$$

With α=1/2 and m = $2m_+/\epsilon$

$$Pr[S_m \geq m_+] \leq e^{-m_+/4}$$

And this probability this bad even occurs with probability at most $=\delta/2$. Which is

$$e^{-m_+/4} \leq \delta/2$$

Solving for m_+ yields,

$$m \geq 4ln(2/\delta)$$

Hence, m$\geq \{\frac{2m_+}{\epsilon}, \frac{4}{\epsilon}ln(2/\delta)\}$
The same analysis holds for bounding the probability that there are fewer than m negative examples in q calls to EX. Hence, m$\geq \{\frac{2m_-}{\epsilon}, \frac{4}{\epsilon}ln(2/\delta)\}$ Combining we get, Hence, m$\geq \{\frac{2m_+}{\epsilon}, \frac{2m_-}{\epsilon}, \frac{8}{\epsilon}ln(\frac{2}{\delta})\}$

Using Chernoff bounds, we could show that drawing a polynomial number of examples in $1/\epsilon$, $1/\delta$ is enough to show that m $\geq$ max$\{m_-$,$m_+\}$ with higher confidence.

case(b.) D is biased towards negative or positive examples which means that $Pr[error_D(h)] \leq \epsilon$

Taken with the cases above we have shown that 2 Oracle PAC learnable is equivalent to standard one-oracle model.

3 Properties of VC Dimension

a. Monotonicity of VC Dimension, if $H^{'} \subseteq H$ then $VCdim(H^{'}) \leq VCdim(H)$

The intuition behind this property of VC Dimension is that every hypotheses that belongs to $H^{'}$ are also in H, so the shattered subset of H is atleast as of $H^{'}$.

Definition: Let H be a finite class. Then, clearly, for any set C we have $|H_C| \leq$ |H| and thus C cannot be shattered if |H| $<$ 2|C|. This implies that VCdim(H) $\leq log_2(|H|)$.

Applying the above definition to hypothesis class $H^{'}$,VCdim($H^{'}$) $\leq log_2(|H^{'}|)$.Applying the above definition to hypothesis class H,VCdim(H) $\leq log_2(|H|)$.
But we know that since, $H^{'} \subseteq H$, which implies, $|H^{'}| \leq |H|$.

Applying log on both sides and since log is an increasing sequence, we have $log_2(|H^{'}|) \leq log_2(|H|)$.

From the above definition, we get

$$VCdim(H^{'}) \leq VCdim(H)$$

.

b. VC Dimension versus log of class size

i. Example of a class H of functions over the real interval X=[0,1] such that H is infinite while VCdim(H)=1

If H is a class of threshold functions where,

$$H_{Threshold} = \{h_x : x \in R\}$$

where,

$$h_x(y) = 1 if y \leq x \quad and \quad 0 \quad otherwise$$

The VCdim($H_{Threshold}$) =1 over the real interval X=[0,1]

ii. Example of a class H of functions over X=[0,1] such that H is infinite while VCdim(H)=$log_2(|H|)$

If H is a class of Parity functions where, For a set I $\subseteq$ {1,2,...,n} we define a parity function, $h_i : \{0,1\}^n \rightarrow \{0,1\}$ as follows: On a binary vector x=$(x_1, x_2, ...x_n) \in \{0,1\}^n$ we have,

$$h_i(x) = (\sum_{i \in I} x_i) \quad mod \quad 2$$

the VC-dimension of the class of all such parity functions, $H^n_{parity} = \{h_I : I \subset \{1, 2, ...n\}\}$ is VCdim(H_{parity})=$log_2(|H|)$

c. VC Dimension of union

i.Prove that VCdim($\cup_{i=1}^r H_i$) $\leq 4dlog(2d) + 2log(r)$

If d=VCdim(H) then,

$$g_H(m) \leq \sum_{i=0}^{d} \binom{m}{i} = \Phi_d(m)$$

$$\begin{aligned} \Pi_{H_1 \cup H_2 ... H_r} &\leq \Pi_{H_1} + \Pi_{H_2} + ... + \Pi_{H_r} \\ &\leq \sum_{i=0}^{d} \binom{m}{i} + \sum_{i=0}^{d} \binom{m}{i} ..... \\ &\leq r(\frac{ek}{d})^d \qquad (1) \end{aligned}$$

From the fact that if m $\leq$ d then $\Pi_c(m) = 2^m$

From the fact that if m > d then $\Pi_c(m) \leq (\frac{ek}{d})^d$

$$\Pi_{H_1 \cup H_2 ... H_r} \leq rk^d$$

Since, the union class can produce all possible labellings on the examples which at most is 2^k. Since, VCdim < $\Pi_{H_1 \cup H_2 ... H_r}$

$$2^k < rk^d$$

Using Lemma, let $a \geq 1$ and b<0 Then $x \geq 4alog(2a) + 2b \Rightarrow x \geq alog(x) + b$

$$rk^d \geq 4dlog(2d) + 2log(r)$$

Combining the equations,

$$2^k \leq 4dlog(2d) + 2log(r)$$

Hence, VCdim($\cup_{i=1}^r H_i$) $\leq 4dlog(2d) + 2log(r)$

ii. Prove that for r=2 it holds that $VCdim(H_1 \cup H_2) \leq 2d+1$

If d=VCdim(H) then,

$$g_H(m) \leq \sum_{i=0}^{d} \binom{m}{i} = \Phi_d(m)$$

From Sauer's Lemma,

$$\begin{aligned}
\Pi_H(2d+c) &\leq \sum_{i=0}^{d} \binom{2d+c}{i} \\
&= 1 + \sum_{i=1}^{d} \binom{2d+c}{i} \\
&= 1 + \sum_{i=1}^{d} \binom{2d+c-1}{i} + \sum_{i=1}^{d} \binom{2d+c-1}{i-1} \\
Since, \binom{n+1}{k} &= \binom{n}{k-1} + \binom{n}{k} \\
&= \sum_{i=1}^{d} \binom{2d+c-1}{i} + \sum_{j=0}^{d} \binom{2d+c-1}{j} \\
&= \sum_{i=1}^{d} \binom{2d+c-1}{i} + \sum_{j=d+c-1}^{2d+c-1} \binom{2d+c-1}{j} \\
&\leq \sum_{i=1}^{d} \binom{2d+c-1}{i} + \sum_{j=d-1}^{2d+c-1} \binom{2d+c-1}{j} \\
&= -1 + 2^{2d+c-1} \\
\Pi_{H_1 \cup H_2}(2d+c) &\leq \Pi_{H_1}(2d+1) + \Pi_{H_2}(2d+1) \\
&\leq 2^{2d+c} - 2 \\
&\leq 2^{2d+c}
\end{aligned} \tag{2}$$

Thus VCdim($H_1 \cup H_2$) cannot be greater than 2d+c.Here, c=1 .
Hence, $VCdim(H_1 \cup H_2) \leq 2d+1$

4

Assume that A' be the online learning algorithm with a mistake bound of m and let S_m be the number of examples needed for the PAC learning algo-

rithm. Let $0<\epsilon<1$ and $0<\delta<1$ .Let the hypothesis set be H. We propose a PAC learning algorithm A as below:
Run online algorithm $A^{'}$ using a new set of examples from EX(c,D) at every stage. If $A^{'}$'s hypothesis h$\in$H is unchanged (no mistake) for the consecutive examples then stop and output h.
For i=1 to m+1:
i. Get the examples from Oracle of batch size s_i.
ii. Use the s_i examples to the online algorithm,$A^{'}$ sequentially
iii. If the hypothesis h_i is wrong on an example,then stop and restart with next iteration i
iv. If the hypothesis continues without any mistake for s_i examples the output the h_i as the solution.
If this algorithm reaches iteration i = (M + 1) then it is guaranteed to stop at this iteration and the current hypothesis will have zero error.

Lets begin by defining the batch size. Assume, for the first iteration a batch size of $2^1 = 2$ examples. If the hypothesis fails for any of these consecutive examples, we increase our batch size to iteration=2 and $2^2 = 4$ examples. Generalizing the series we get the batch size to be 2^i, where i is the number of the iteration.
Now we need to prove that the assumption works for the PAC learning algorithm as well.

$$\begin{aligned} Pr[PAC - Learnerfails] &\leq \sum_{i=1}^{m+1} Pr[output \quad hyp \quad with \quad err > in \quad the \quad i^{th} iteration] \\ &\leq \sum_{i=1}^{m+1} \frac{\delta}{2^i} \\ &\leq \delta \sum_{i=1}^{m+1} \frac{1}{2^i} \\ &\leq \delta \end{aligned} \tag{3}$$

Hence, the maximum error is bounded still by δ.

Consider some fixed hypothesis h which the algorithm constructs. Suppose that h is bad, Pr[h(x)=c(x)]<(1-ϵ)
Pr[the bad hypo is correct on m_i examples in a row] $\leq (1-\epsilon)^{m_i}$ (By applying a Union Bound)

We are allowed a maximum of $\delta/2^i$ inconsistent hypothesis in each iteration. For the i^{th} iteration, if h is a bad hypothesis then we can say that the maximum probability that our hypo for the i^{th} iteration is wrong is also less than $\delta/2^i$.
So,

$$\begin{aligned} Pr[h \quad is \quad wrong] &\leq \delta/2^i \\ (1-\epsilon)^{m_i} &\leq \delta/2^i \\ e^{-\epsilon.m_i} &\leq \delta/2^i \\ Since (1+x) &\leq e^x \\ \text{Taking log on both sides,} \\ -\epsilon.m_i &\leq log(\delta/2^i) \\ \text{Solving we get,} \\ m_i &\geq \frac{1}{\epsilon} log \frac{2^i}{\delta} \end{aligned} \tag{4}$$

Which gives us the minimum number of examples for each iteration. Also, we know that m is the total number of examples given to the algorithm. Hence,

$$\begin{aligned} m &\leq \sum_{i=1}^{m+1} m_i \\ &= \sum_{i=1}^{m+1} \frac{1}{\epsilon} log \frac{2^i}{\delta} \\ &= \sum_{i=1}^{m+1} \frac{1}{\epsilon} [ilog2 + log(\frac{1}{\delta})] \\ &= O(\frac{1}{\epsilon} m^2 + \frac{1}{\epsilon} mlog(\frac{1}{\delta})) \end{aligned} \tag{5}$$

Hence, the sample complexity for our PAC algorithm is $O(\frac{1}{\epsilon}m^2 + \frac{1}{\epsilon}mlog(\frac{1}{\delta}))$.

5

Let X be the infinite set 1,2,3,.... $P_1, P_2, ...$ be an infinite list of computer programs. Each program takes an input x $\in$X, runs a functions $f_i : X \rightarrow \{0, 1\}$ and outputs {0,1}. Assuming, the computer program is enumerated, such that there exists a computer program M which given an input value i, outputs P_i.

To find: a learning algorithm which is guaranteed to make $O(\log$ t) prediction mistakes where t is the smallest index such that f=f_t.

To find the learning algorithm which makes $O(\log$ t) mistakes, we will be using the Halving algorithm. In the current context, suppose that we (the player) have access to the predictions of N "programs."

Algorithm:

Maintain a list of programs which have not yet made mistakes, initially including all programs.

On each trial, Make prediction based on majority vote of programs in list:

$$y^{'} = 1 \quad if|\{i : h_i(x) = 1\}| > |\{i : h_i(x) = 0\}|, else \quad 0$$

Eliminate experts i in list which made mistakes $h_i(x) \neq y$

To analyze the Halving Algorithm, let W = Number of surviving programs (correct programs left in list). Initially, W = N. If the learner makes a mistake, then $\geq 1/2$ of the remaining experts are eliminated since the learner makes a prediction based on a majority vote of the programs. So,

$$\begin{aligned} After \quad 1 \quad mistake, \quad W \leq 1/2(N) \\ After \quad 2 \quad mistakes, \quad W \leq 1/2(N) \\ ... \\ After \quad m \quad mistakes, \quad W \leq 2^{-m}(N) \end{aligned} \tag{6}$$

From the question, we are guaranteed that there is one function f which is perfect and will never be thrown out, so W $\geq$ 1. Therefore, we have 1$\leq$ W $\leq 2^{-m}(N)$, which implies the learner makes m $\leq$ log(N) mistakes.

Analysis:

To bound the number of mistakes without knowing the size of the list of programs, let us assume a highly increasing function for the number of programs (or the number of examples) of the form $2^{2^{(i-1)}}$. The intuition behind the proof is: Say, if we don't find the f_t in a set of programs(or examples), then we need to look at more programs(or examples) to find the f_t.

For the increasing function $2^{2^{(i-1)}}$ which generates the number of examples. Initially, For i=1, there will be 2 examples and the maximum number of mistakes will be 1, since according to the halving algorithm if m is the number of examples, then there can be atmost log(m) mistakes. For i=2, there will be 4 examples and atmost 2 mistakes. Generalizing, assuming that the element in the sequence before the last is t, then the last element will be t^2 and the number of mistakes will be atmost log(t^2)=2log(t).

Number of Examples	Number of mistakes
2	1
2^2	2
2^{2^2}	2^2
2^{2^3}	2^3
.	.
.	.
.	.
t^2	$2log(t)$

To get a bound on the number of prediction mistakes, we have to sum the number of mistakes till t^2 number of examples. Note that, the number of mistakes is increasing in a geometric progression. To find the n^{th} term (which is $2log(t)$)in a geometric progression we use the below formula,

$$t_n = t_1.r^{n-1}$$

$$2log(t) = 1.2^{n-1}$$

Taking log on both sides,

$$log(2log(t)) = n - 1$$

$$n = log(2log(t)) + 1$$

To find the summation of the geometric sequence,

$$S_n = a(r^n - 1)/(r - 1)$$

Here a=1,r=2 and substituting n,

$$S_n = 1(2^{log(2log(t))+1} - 1)/(2 - 1)$$

$$S_n = (4log(t) - 1)$$

Taking the upper bound on the sum, we have that $O(logt)$ number of prediction mistakes. Hence, we used the halving algorithm and showed it makes $O(\log \text{t})$ mistakes.