**ML Foundation HW2**

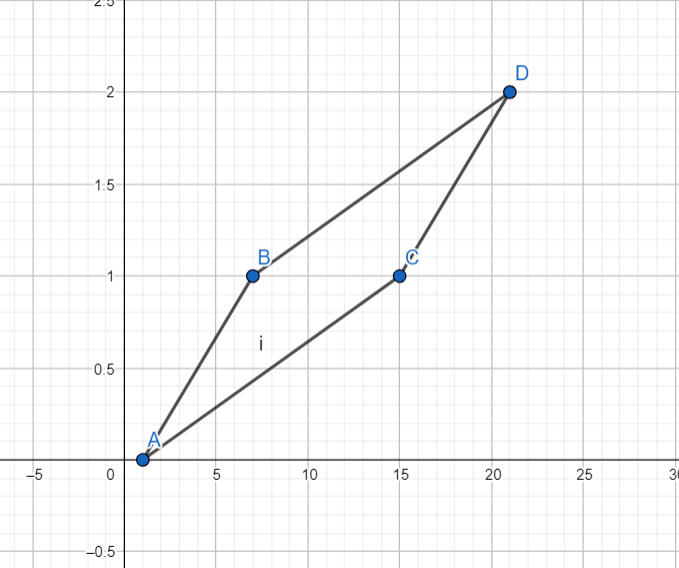
**1. C**

這題我們不討論基礎現代計算，所以我可能會直接列出基底及dimension(如果要的話，找一個matrix calculator也能做一樣的事情)。

**( a )**

注意他們在同一條直線(i.e. dim(span)=1)，所以很類似positive ray的情況，沒辦法shatter。

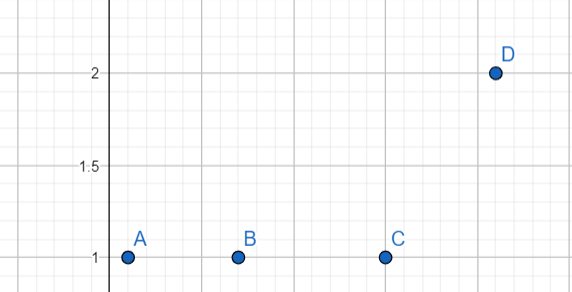
**( b )**

他們的span的dimension為2，以(1,1,1),(0,1,2)(1,1,1),(0,1,2)當基底的話，轉換基底結果他們座標分別是(1,0),(7,1),(15,1),(21,2)(1,0),(7,1),(15,1),(21,2)，注意轉換基底後(a,b)=a⋅(1,1,1)+b⋅(0,1,2)(a,b)=a⋅(1,1,1)+b⋅(0,1,2)。因為他們在同一平面，所以一個在R3R3的平面在此面截出的線即足夠判斷他們的分類關係。  
我們在R2R2畫出他們：  
  
如果我們設A=D=+1,B=C=−1A=D=+1,B=C=−1，找不到一條在R2R2的線分開他們，也們辦法在R3R3找出一平面分開他們。

**( c )**

(1,3,5)(1,3,5)跟其他人不在同平面，(7,8,9),(15,16,17),(21,23,25)(7,8,9),(15,16,17),(21,23,25)在同平面(稱為EE)且不在同線上，所以我們對這三點，任何yy的組合都可找到線段把他們三個shatter，接下來回到R3R3的情況，可以找到一簇平面使得他們跟EE的截線相等，所以只要轉一下平面，就可以找到兩個平面把(1,3,5)(1,3,5)分成+1+1或−1−1。所以每一組yy們都可以找到平面把他們分類，所以這組可被shatter。

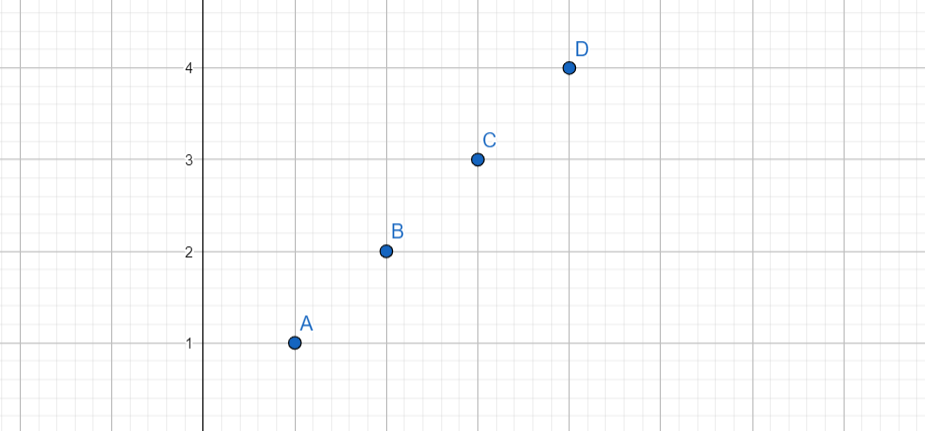
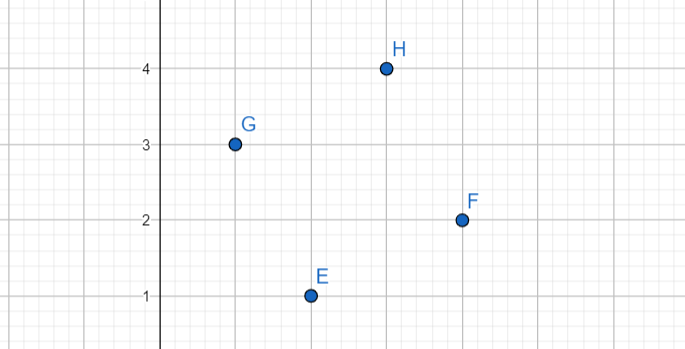
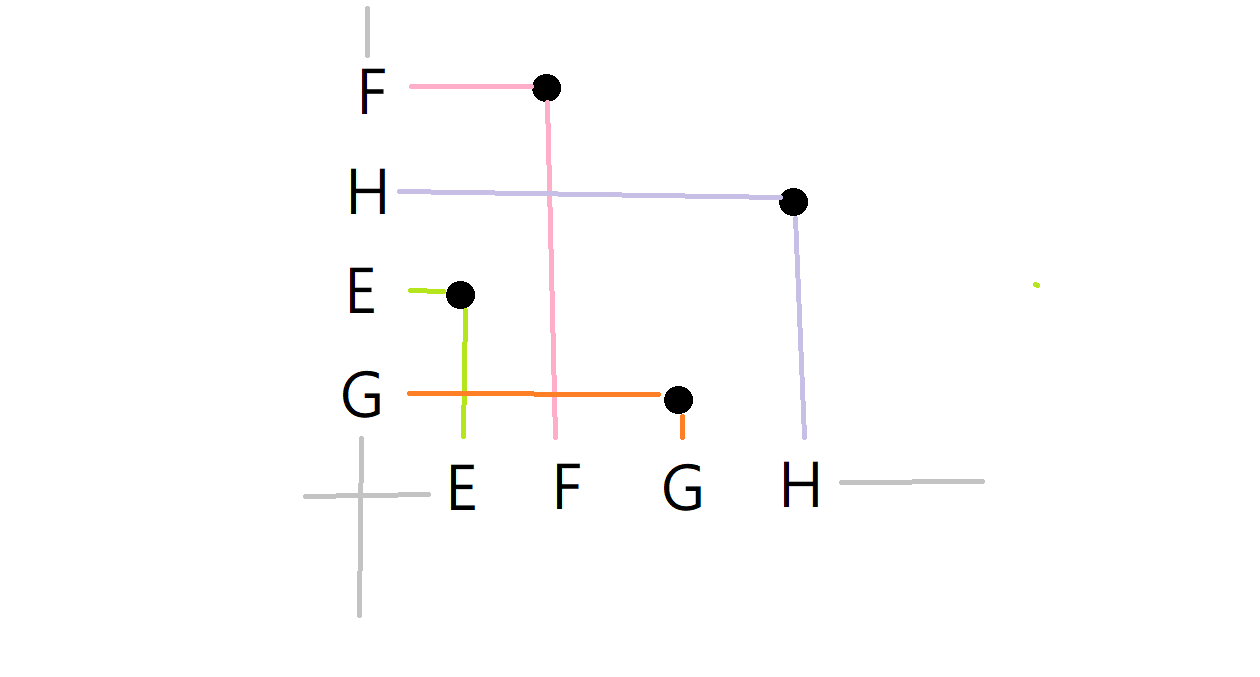
**( d )**

他們在同平面，用基底(1,1,1),(0,1,2)(1,1,1),(0,1,2)，他們會被轉換基底成(1,1),(7,1),(15,1),(21,2)(1,1),(7,1),(15,1),(21,2)：  
  
跟( a )的狀況一樣，A=C=+1,B=D=−1A=C=+1,B=D=−1沒辦法被分開

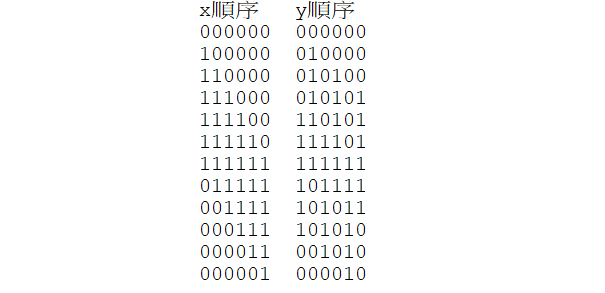
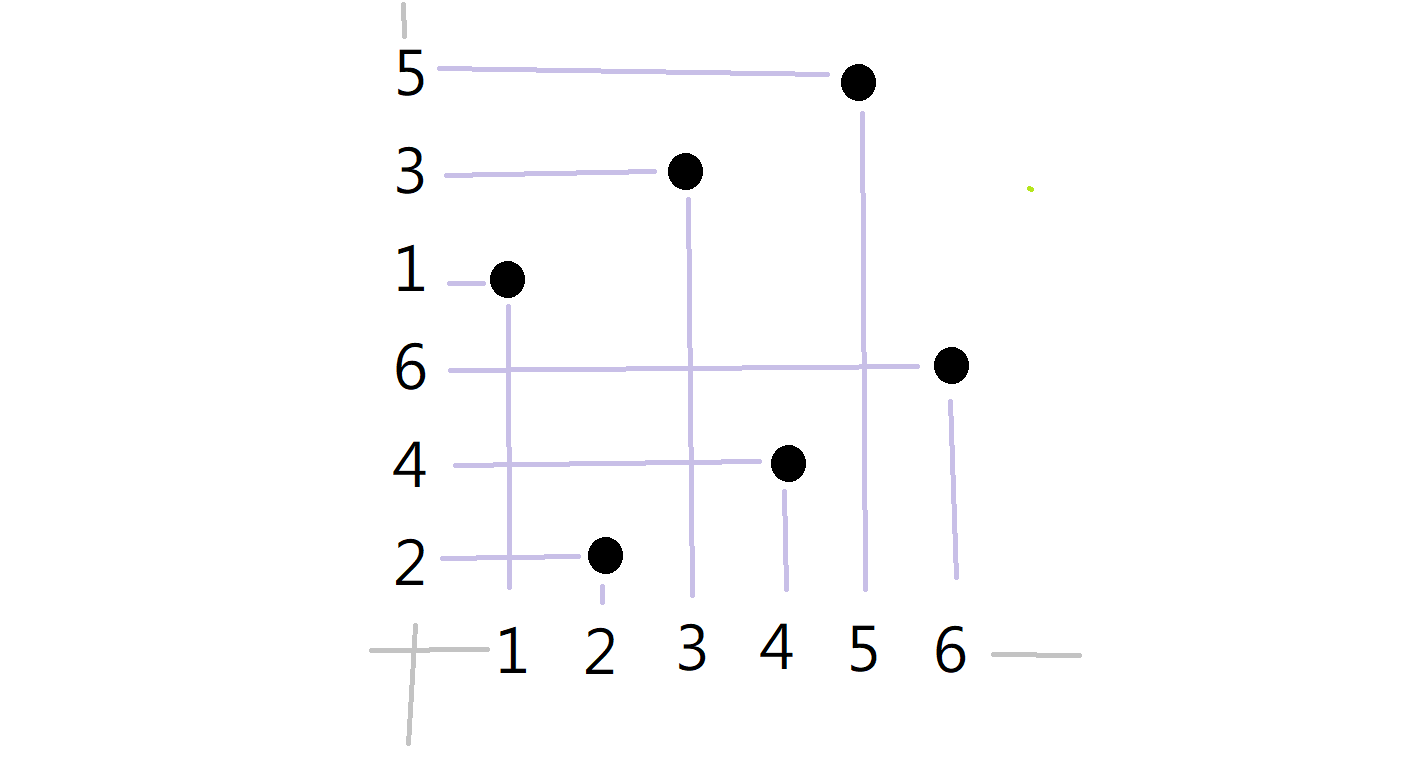
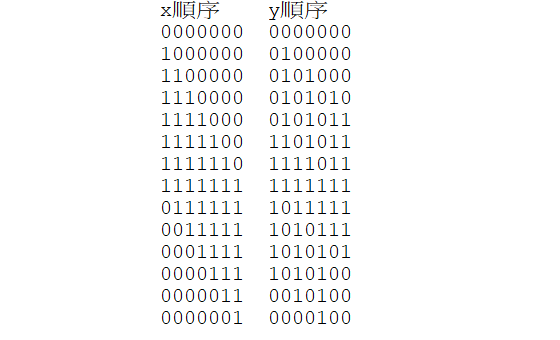
**( e )**

他們在同平面，用基底(1,1,1),(0,1,2)(1,1,1),(0,1,2)，他們會被轉換基底成(1,1),(4,1),(7,1),(15,1),(21,2)(1,1),(4,1),(7,1),(15,1),(21,2)，基本上跟( a )和( d )一樣的邏輯，存在yy們使(1,1),(4,1),(7,1)(1,1),(4,1),(7,1)沒法被分開。

**2. D**

我們參考1D positive ray的情況，可以發現對於每一組資料點，分類狀況是被他們「在x與y軸的排列順序」所限制的。(這裡用y表y軸，yy表「某個hypothesis對某組資料點們得出的值/prediction」。)  
舉例，如果我們有這組資料點：  
  
唯一決定他們能有多少yy的組合的只有他們在x與y軸的順序。在這個例子裡x軸從左到右的順序為ABCD，從下到上的順序同為ABCD。  
所以對於y=[yA,yB,yC,yD]y=[yA,yB,yC,yD]，我們知道如果用橫線(i.e. y順序)來對他們分類的話，yy有[0000][1000][1100][1110][1111][0111][0011][0001][0000][1000][1100][1110][1111][0111][0011][0001]這八種可能(我用1,01,0表示分類，跟其他地方的+1,−1+1,−1一樣的概念)。  
注意用縱線(i.e. x順序)同樣只能做出這幾種可能(因為跟用橫線分類能夠做出的yy撞到了)。所以這個例子裡，只能做出8種yy，因為x順序和y順序可以做出相同的組合，不是最好的排列法。  
我們看另一個例子：  
  
他的x順序為GEHF，y順序為EFGH。對於y=[yE,yF,yG,yH]y=[yE,yF,yG,yH]，用y順序(i.e. 用橫線分類)可以做出[0000][1000][1100][1110][1111][0111][0011][0001][0000][1000][1100][1110][1111][0111][0011][0001]這八種。用x順序(i.e. 用縱線分類)可以做出[0010][1010][1011][0100][0101][1101][0010][1010][1011][0100][0101][1101]這六種。注意不論是用x還是y順序都可以做出[1111][1111]和[0000][0000]，只能算一次。  
我們可以知道給定有nn項的x或y順序，可以做出2n2n種yy。如果我們能夠避免x及y順序做出一樣的yy(如第二例)，我們最多能得到4n−24n−2種組合(-2是因為[000...0][000...0]和[111...1][111...1]必相撞，扣除其中一組)。注意給定任意x和y順序，我們都能找到對應的資料點座標：  
  
(x,y軸反了不過你懂的。)  
所以growth function最大為4n−24n−2。

事實上，對於任意n>4n>4，我們應該可以直接製造出能做出4n−24n−2種yy的x順序與y順序才對。  
我們對點編號，從最左邊到最右邊從1編號到N，所以x順序為1234…N(數字為編號)。  
對於任意偶數nn，可能可以用y順序246…135…(先填偶數再填奇數)，應該都能做出4n−24n−2種yy。對於任意奇數nn，可以用y順序123…(N-1)N123…(N-2) (例如n=9n=9，y順序246891357)。下面的程式可以輸入nn並給出此種x, y順序做出的所有可能性，供參考：  
[(https://www.csie.ntu.edu.tw/~b08902068/ML\_HW/ML\_HW2\_prob2.py)](https://www.csie.ntu.edu.tw/~b08902068/ML_HW/ML_HW2_prob2.py)  
我們接下來以直觀方式證明對每個n都真的可用此種方法做出4n−24n−2種yy：

我們用n=6n=6與n=7n=7的狀況來直觀證明。  
對於n=6n=6，x順序為123456，y順序為246135。可知會做出下列yy：  
  
  
可以發現，x順序都是做出連續的1。注意100000與000001的部分，事實上就是這兩個yy使我們討論奇數NN的時候要刻意移動N，因為N在尾端的話就會跟這兩個其中一個撞到了。  
注意我們做y順序的方式是每次隔一格加1，因為x順序做出的yy的1都是連續的，這樣不連續的做導致中間部分不可能會跟x相撞，除了111111。  
所以只要看兩個末端的edge case有沒有撞到即可。因為y順序是從2開始填，所以不會跟x順序撞。最後一個000010也不會跟x順序撞。基本上所有偶數都是差不多的狀況，在此當作證畢。  
至於n=7n=7，x順序為1234567，y順序為2467135：  
  
可以發現他的y順序的edge case因為刻意移動N，所以不會做出如0000001等會跟x順序撞到的yy(其實如果y順序為2461357的話就會撞了)。中間因為也是隔一格填所以都不會撞到x順序。當作奇數證畢。  
所以的確，對任意n≥4n≥4，都可找到一組資料，使之可以做出4n−24n−2組yy。

因為上面引入x順序和y順序的契機就是因為只有這兩個東西影響，並且我們上面也說明了最多也只能做出4n−24n−2種yy，我們接下來又找出真的可以做出4n−24n−2種yy的x和y順序的方法，所以基本上已經沒有其他可能了，growth function不多不少就是4n−24n−2。QED.

**3. C**

我們證明N=2N=2會被shatter：

考慮w0=1,x1=[1,0,1],x2=[1,1,0]w0=1,x1=[1,0,1],x2=[1,1,0]，對於所有yy的可能性都能找到ww，所以可shatter：

* y=[−1,−1]⇒w=[1,−3,−3]y=[−1,−1]⇒w=[1,−3,−3]
* y=[−1,+1]⇒w=[1,−3,+3]y=[−1,+1]⇒w=[1,−3,+3]
* y=[+1,−1]⇒w=[1,+3,−3]y=[+1,−1]⇒w=[1,+3,−3]
* y=[+1,+1]⇒w=[1,+3,+3]y=[+1,+1]⇒w=[1,+3,+3]

對所有可能的yy都存在ww，所以被shatter。

我們以下證明N=3N=3的狀況無法被shatter：

我們假設資料點xi=[1,xi1,xi2]∈R3xi=[1,xi1,xi2]∈R3，且定義xi=[xi1,xi2]∈R2xi=[xi1,xi2]∈R2。可以想成xi=[1,xi]∈R3xi=[1,xi]∈R3，因為xx的第0維度一定是1所以這樣寫可以想成xixi是真的拿到的，增維之前的資料點。  
我們的資料點為x1,x2,x3x1,x2,x3。  
現在證明對任何資料組(N=3N=3)，無法shatter。

我們知道x1,x2,x3x1,x2,x3的x1,x2,x3∈R2x1,x2,x3∈R2，必線性相依。  
WLOG，我們設x3=a⋅x1+b⋅x2x3=a⋅x1+b⋅x2。  
設w=[w0,w1,w2]w=[w0,w1,w2]，定義w=[w1,w2]∈R2w=[w1,w2]∈R2。我們知道如果yi=+1yi=+1的話，wTxi=w0+wTxi>0⇒wTxi>−w0wTxi=w0+wTxi>0⇒wTxi>−w0，同理，如果yi=−1yi=−1的話，wTxi<−w0wTxi<−w0。並且wTx3=a⋅wTx1+b⋅wTx2wTx3=a⋅wTx1+b⋅wTx2。  
我們假設a>0,b>0a>0,b>0，假設y=[−1,−1,y3]y=[−1,−1,y3]：  
wTx3<−(a+b)w0=a⋅wTx1<−aw0+b⋅wTx2<−bw0wTx3⏟<−(a+b)w0=a⋅wTx1⏟<−aw0+b⋅wTx2⏟<−bw0  
所以只要a+b>1a+b>1，就不可能做得出y=[−1,−1,+1]y=[−1,−1,+1]。  
假設y=[+1,+1,y3]y=[+1,+1,y3]：  
wTx3>−(a+b)w0=a⋅wTx1>−aw0+b⋅wTx2>−bw0wTx3⏟>−(a+b)w0=a⋅wTx1⏟>−aw0+b⋅wTx2⏟>−bw0  
所以只要a+b<1a+b<1，就不可能做得出y=[+1,+1,−1]y=[+1,+1,−1]。如果a+b=1a+b=1，不論是怎麼定義sign(0)sign(0)的，其中一方必定出錯。我們證明出不論a,b>0a,b>0為多少，都不能shatter。  
接下來只要證所有正負號可能即可。老實說四種可能性的證明相似，我們接下來只再證a>0,b<0a>0,b<0的情況，其他兩種可以根據已證明的這兩種狀況同理而證。  
假設a>0,b<0a>0,b<0，我們看y=[−1,+1,y3]y=[−1,+1,y3]的情況。可知道：  
wTx3<−(a−|b|)w0=a⋅wTx1<−aw0+b⋅wTx2<|b|w0wTx3⏟<−(a−|b|)w0=a⋅wTx1⏟<−aw0+b⋅wTx2⏟<|b|w0  
注意因為b<0b<0，wTx2>−w0⇒b⋅wTx2<−b⋅w0⇒b⋅wTx2<|b|⋅w0wTx2>−w0⇒b⋅wTx2<−b⋅w0⇒b⋅wTx2<|b|⋅w0。可證出a−|b|>1a−|b|>1的話做不出y=[−1,+1,+1]y=[−1,+1,+1]。同理，a−|b|<1a−|b|<1的話做不出y=[+1,−1,−1]y=[+1,−1,−1]。注意a−|b|=a+ba−|b|=a+b。  
所以同理可證對所有a,ba,b，根據a+ba+b跟1的比較，一定做不出y=[−sign(a),−sign(b),+1]y=[−sign(a),−sign(b),+1]以及y=[sign(a),sign(b),−1]y=[sign(a),sign(b),−1]其中一種。QED。

所以dVC=2dVC=2。

**4. B**

我們僅考慮任何點跟原點的距離(注意所有距離均大於等於0)，會發現這題跟positive region一模一樣。所以(N+12)+1(N+12)+1。

**5. B**

(N+12)+1=O(n2)(N+12)+1=O(n2)，所以dVC=2dVC=2。或是可以輕易靠positive region的經驗知道N=2N=2可被shatter但N=3N=3不行。

**6. D**

我們假設∀h∈H:|Ein(h)−Eout(h)|≤ϵ∀h∈H:|Ein(h)−Eout(h)|≤ϵ成立，我們知道：  
{Eout(g)≥Eout(g∗)Ein(g)≤Ein(g∗){Eout(g)∈[Eout(g)−ϵ,Eout(g)+ϵ]Ein(g∗)∈[Ein(g∗)−ϵ,Ein(g∗)+ϵ]{Eout(g)≥Eout(g∗)Ein(g)≤Ein(g∗){Eout(g)∈[Eout(g)−ϵ,Eout(g)+ϵ]Ein(g∗)∈[Ein(g∗)−ϵ,Ein(g∗)+ϵ]  
所以我們知道：  
Eout(g)−Eout(g∗)≥0≤(Ein(g)+ϵ)−(Eout(g∗)−ϵ)≤Ein(g)−Ein(g∗)≤0+2⋅ϵ≤2⋅ϵEout(g)−Eout(g∗)⏟≥0≤(Ein(g)+ϵ)−(Eout(g∗)−ϵ)≤Ein(g)−Ein(g∗)⏟≤0+2⋅ϵ≤2⋅ϵ  
既然我們找出最大上限為Eout(g)−Eout(g∗)≤2⋅ϵEout(g)−Eout(g∗)≤2⋅ϵ，那只要找到ϵϵ就好了。  
我們設剛剛的假設成立的機率至少為1−δ1−δ，那可知我們需要此假設不成立的機率小於δδ，意即：  
Pr(∃h∈H:|Ein(h)−Eout(h)|>ϵ)≤δPr(∃h∈H:|Ein(h)−Eout(h)|>ϵ)≤δ  
且δδ應為upper bound。我們從定理可得知另一個upper bound，為了保證成立，δδ應為更強的upper bound，所以：  
4mH(2N)exp(−18ϵ2N)≤δexp(−18ϵ2N)≤δ4mH(2N)−18ϵ2N≤ln(δ4mH(2N))ϵ≥√−8Nln(δ4mH(2N))=√8Nln(4mH(2N)δ)4mH(2N)exp(−18ϵ2N)≤δexp(−18ϵ2N)≤δ4mH(2N)−18ϵ2N≤ln⁡(δ4mH(2N))ϵ≥−8Nln⁡(δ4mH(2N))=8Nln⁡(4mH(2N)δ)  
所以2⋅ϵ≥2⋅√8Nln(4mH(2N)δ)2⋅ϵ≥2⋅8Nln⁡(4mH(2N)δ)為Eout(g)−Eout(g∗)Eout(g)−Eout(g∗)的upper bound。

**7. D**

因為一組資料最多也只能做到對所有hh的輸出都不同，所以在⌊log2M⌋⌊log2⁡M⌋個資料點的情形下說不定還可以被shatter(2⌊log2M⌋≤M2⌊log2⁡M⌋≤M)，但是超過的話一定不可能shatter(因為最大可能性也只有M組y)。所以dVCdVC最高為⌊log2M⌋⌊log2⁡M⌋。

**8. D**

對一組D，我們知道對N=k+1N=k+1的資料是能shatter的。只要這kk個資料點的1的數目不同(從0個到k個)，他們就互相independant。如果是N=k+2N=k+2的話，根據鴿籠原理我們知道一定有兩個資料點，他們的1的個數相同。所以我們不可能使這兩個點的y不同，導致不能shatter。  
所以dVC=k+1dVC=k+1。

**9. C**

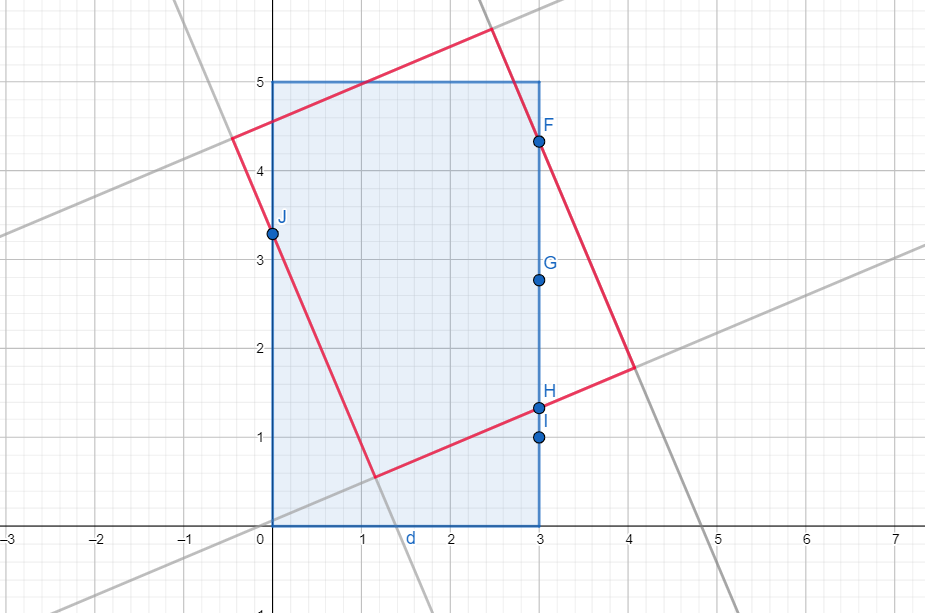
他需要：

* Some set of d distinct inputs is shattered by H
* Any set of d+1 distinct inputs is not shattered by H
* Some set of d + 1 distinct inputs is not shattered by H

所以C。

**10. C**

**( a )**

首先，如果要shatter的話，其中一個yy必須讓所有點的h(x)h(x)都為0，代表所有點會在一長方形的四個邊上面。  
但是，假設這個長方形的某一邊有超過四個點的話，會有不能shatter的情況。我們假設他們所在的長方形的邊平行x和y軸(WLOG，旋轉即可推廣到其他長方形)。我們假設某個縱邊有四個點，由上往下編號FGHI。我們會知道其實是可以任意挑其中兩個點被分成0的(i.e. h(x)=0h(x)=0)，因為只要hypothesis的長方形歪一邊，即可任意挑兩個點經過，請看FGHI點：  
  
即使中間有隔點G也沒關係，還是有hypothesis可以只碰到F和H。  
不過現在假設我們給這FGHI點的yy為[0010][0010]，也就是長方形要通過FGI點卻不碰到H點。  
我們知道hypothesis長方形的邊都不能剛好落在FGHI線上，因為這會變成類似positive region的情形，且我們知道positive region沒辦法分出這種yy(因為被判斷為0的點不連續出現)。我們也知道一長方形被任意線截只能截出兩點(如果此線不平行邊的話)，所以不能像上面那麼幸運，轉個方向就能做出過一線上三點的hypothesis長方形。所以是找不出hypothesis長方形可以剛好過FGI點的，代表一邊出現4個點以上就找得到yy沒辦法做出來。  
所以我們知道，只要一邊有4個點以上就不能shatter，又長方形四個邊最多容納4⋅3=124⋅3=12個點，假設真的能shatter 12個點的情況好了，dVCdVC最高也才12。注意如果有點落在長方形頂點上不會更好只會更糟，因為這個點會同時屬於兩個邊。  
所以我們證明出dVC≤12<∞dVC≤12<∞。

**( b )**

因為兩個區間的交集必為一區間(或空集合)，所以把所有區間聯集起來後，就變成一個區間的版本(i.e. h(x)=+1 if x in (a,b))。這題其實就變成positive region了，dVC=2dVC=2。

**( c )**

(參考：<https://cs.nyu.edu/~mohri/ml16/sol2.pdf>, 第五頁。是從袁紹奇那邊聽到這個的。)  
對於給定的nn，我們找到一組點{(xi,yi)|i∈N,i∈[1,n],xi=2−i}{(xi,yi)|i∈N,i∈[1,n],xi=2−i}，使得可以被這組hypothesis shatter。意即，dVC=∞dVC=∞。

我們找到w=π(1+∑ni=12iy′i)w=π(1+∑i=1n2iyi′)，且y′i=(1−yi)/2yi′=(1−yi)/2 (意即，yi=+1→y′i=0,yi=−1→y′i=1yi=+1→yi′=0,yi=−1→yi′=1。)  
我們可以導出：  
wxj=π(1+n∑i=12iy′i)⋅2−j=π(2−j+n∑i=12i−jy′i)=π(2−j+j−1∑i=12i−jy′i+y′j+n∑i=j+12i−jy′i)wxj=π(1+∑i=1n2iyi′)⋅2−j=π(2−j+∑i=1n2i−jyi′)=π(2−j+∑i=1j−12i−jyi′+yj′+∑i=j+1n2i−jyi′)  
注意在括號裡的第四項裡，i−j>0i−j>0，又y′i∈{0,1}⊆Zyi′∈{0,1}⊆Z，所以∑∑裡的每一項都會是22的倍數，配上括號外乘的ππ，我們知道這一個∑∑的結果是2π2π的倍數，因為sinsin的週期性，等等考慮sinsin值的時候直接不考慮這一項也可以。  
又我們知道∑j−1i=12i−jy′i∑i=1j−12i−jyi′裡，y′i∈{0,1}yi′∈{0,1}，所以∑j−1i=12i−jy′i≤∑j−1i=12i−j=2−1+2−2+...+2−j∑i=1j−12i−jyi′≤∑i=1j−12i−j=2−1+2−2+...+2−j，代表加上第一項的2−j2−j後小於等於1。  
所以最後的式子是：π(2−j+y′j+2⋅(...))≤wxj≤π(1+y′j+2⋅(...))π(2−j+yj′+2⋅(...))≤wxj≤π(1+yj′+2⋅(...))  
假設yj=+1,y′j=0yj=+1,yj′=0，那sin(wxj)∈[sin(π⋅2−j),sin(π)]sin⁡(wxj)∈[sin⁡(π⋅2−j),sin⁡(π)]，又2−j<12−j<1，所以sign(sin(wxj))=+1sign(sin⁡(wxj))=+1。注意事實上不需要考慮等於sin(π)sin⁡(π)的情況，因為等號成立代表所有y′i=1,yi=−1yi′=1,yi=−1，但是y′j=0yj′=0，所以不用考慮。  
假設yj=−1,y′j=1yj=−1,yj′=1，那可知sin(wxj)∈[sin(π⋅2−j+π),sin(π+π)]sin⁡(wxj)∈[sin⁡(π⋅2−j+π),sin⁡(π+π)]，又2−j<12−j<1所以sign(sin(wxj))=−1sign(sin⁡(wxj))=−1。同樣不需要考慮等於sin(π+π)sin⁡(π+π)，因為那會代表所有y′i=1,yi=−1yi′=1,yi=−1，我們知道存在hypothesis w=−1w=−1使這組點產生這組yiyi。  
所以這組xixi，對於任意yiyi，的確存在hypothesis ww使他們shatter。

雖然不是我發想的，不過能稍微分享一些發現：  
他非常妥善應用∑2iy′i⋅2−j∑2iyi′⋅2−j，因為可以把各個點分成整數區(2的倍數)、小數區和y′jyj′，又證明了小數區不會超過1，剛好可以當作sinsin的分界。至於y′iyi′只要找個{+1,−1}{+1,−1}到某個集合的mapping且剛好對到我們需要的性質就可以了。  
需要的性質為：

* 必須為0或1，因為才能使小數區不超過1。當然可以事後靠常數倍數調整不過他是在這裡作業。
* 因為y′jyj′項會剛好變成位移，如果想要sinsin值是負的話需要把位移變成ππ。

所以才會有上述的y′i:yi↦(1−yi)/2yi′:yi↦(1−yi)/2。

**( d )**

能做出的yy實際上只有兩種，dVC=1dVC=1。

**11. D**

為方便，定義Eout(h,0)=KEout(h,0)=K。  
我們會知道可以預期抽取的資料有KK比例判斷錯(i.e. [h(x)≠y]=1[h(x)≠y]=1)，有1−K1−K判斷對(i.e. [h(x)≠y]=0[h(x)≠y]=0)。  
又給定ττ，我們知道對[h(x)≠y]=1[h(x)≠y]=1的xx，有1−τ1−τ的機率維持[h(x)≠y]=1[h(x)≠y]=1，有ττ的機率變成00；另一情況同理。  
所以我們知道在PP的xx中：

* 對[h(x)≠y]=1[h(x)≠y]=1的xx比例有KK
  + 在PτPτ判斷錯誤的機率有1−τ1−τ，判斷成功的機率有ττ
* 對[h(x)≠y]=0[h(x)≠y]=0的xx比例有1−K1−K
  + 在PτPτ判斷成功的機率有1−τ1−τ，判斷錯誤的機率有ττ

所以我們知道，總結而言被PτPτ判斷錯誤的點有：  
Eout(h,τ)=K⋅(1−τ)+(1−K)⋅τEout(h,τ)=K⋅(1−τ)+(1−K)⋅τ  
給定ττ，定義Eout(h,τ)=γEout(h,τ)=γ，可知：  
γ=K⋅(1−τ)+(1−K)⋅τ⇒γ=K+τ−2⋅Kτ⇒K(1−2τ)=γ−τ⇒K=γ−τ1−2τγ=K⋅(1−τ)+(1−K)⋅τ⇒γ=K+τ−2⋅Kτ⇒K(1−2τ)=γ−τ⇒K=γ−τ1−2τ  
所以Eout(h,0)=Eout(h,τ)−τ1−2τEout(h,0)=Eout(h,τ)−τ1−2τ。

**12. B**

我們會知道，我們可以把資料分成三種：xixi(i∈{1,2,3}i∈{1,2,3})，且f(xi)=if(xi)=i，那：  
P(y|x1)=⎧⎨⎩0.7,y=10.1,y=20.2,y=3,P(y|x2)=⎧⎨⎩0.7,y=20.1,y=30.2,y=1,P(y|x3)=⎧⎨⎩0.7,y=30.1,y=10.2,y=2,P(y|x1)={0.7,y=10.1,y=20.2,y=3,P(y|x2)={0.7,y=20.1,y=30.2,y=1,P(y|x3)={0.7,y=30.1,y=10.2,y=2,  
我們知道x∈[0,1]3⊆R3x∈[0,1]3⊆R3，因為uniform分布，所以不會對任何一維偏頗，所以f(x)=1,2,3f(x)=1,2,3的機率相等，都為1/31/3。注意因為是對R3R3的子集合做取樣，對於產生如[0.3,0.5,0.5][0.3,0.5,0.5]等有多維同值的狀況機率應趨近於0，所以在這裡就不多考慮f(x)f(x)在xx存在多個argmaxargmax可行的時候的定義，或說怎麼定義都沒差。為了完整論述我們姑且定義argmaxargmax都是取最小的。(i.e. f([0.3,0.5,0.5])=2f([0.3,0.5,0.5])=2)  
既然f(x)f(x)為1,2,31,2,3的機率相等，我們直接計算各狀況的square error，再取平均得Eout(f)Eout(f)的誤差。

* f(x)=1f(x)=1: E(err)=0.7⋅0+0.2⋅1+0.1⋅4=0.6E(err)=0.7⋅0+0.2⋅1+0.1⋅4=0.6
* f(x)=2f(x)=2: E(err)=0.7⋅0+0.2⋅1+0.1⋅1=0.3E(err)=0.7⋅0+0.2⋅1+0.1⋅1=0.3
* f(x)=3f(x)=3: E(err)=0.7⋅0+0.2⋅4+0.1⋅1=0.9E(err)=0.7⋅0+0.2⋅4+0.1⋅1=0.9

所以誤差為13(0.6+0.3+0.9)=0.613(0.6+0.3+0.9)=0.6。

**13. B**

可算出f∗(x1)=1.5,f∗(x2)=1.9,f∗(x3)=2.6f∗(x1)=1.5,f∗(x2)=1.9,f∗(x3)=2.6，可知Ex∼P(x)(f∗(x)−f(x))2=13((1−1.5)2+(2−1.9)2+(3−2.6)2)=0.14Ex∼P(x)(f∗(x)−f(x))2=13((1−1.5)2+(2−1.9)2+(3−2.6)2)=0.14。

**14. D**

我們知道mH(2N)=4NmH(2N)=4N，所以對每個選項算upper bound，得：

| **N** | **Upper bound** |
| --- | --- |
| 6000 | 53.09609953419205 |
| 8000 | 5.811191009598064 |
| 10000 | 0.5962645075325878 |
| 12000 | 0.0587332455363506 |
| 14000 | 0.005624638108866525 |
| 為了δ<0.1δ<0.1，只有D,E符合。選D。 |  |
| 或者說4⋅4N⋅e−0.01N/8≤0.14⋅4N⋅e−0.01N/8≤0.1的解為N>11543.2orN<0.00625005N>11543.2orN<0.00625005(拿wolfram alpha解的結果，注意N>0N>0，所以只考慮前半的不等式)。 |  |

**15. B**

τ=0τ=0，s=+1s=+1，所以h+1,θ=sign(x−θ)h+1,θ=sign(x−θ)。如果大於θθ的話即為+1，反之-1。同時f(x)=sign(x)f(x)=sign(x)。  
假設θ<0θ<0，我們知道[θ,0][θ,0]的地方都會被預測錯(-1預測成+1)；假設θ>0θ>0，我們知道[0,θ][0,θ]的地方都會被預測錯(+1預測成-1)。所以在區間長度22的地方都會有|θ||θ|的段落長度是錯的，所以Eout(h+1,θ)=|θ|/2Eout(h+1,θ)=|θ|/2。  
話說回來，Eout(h−1,θ)Eout(h−1,θ)應該會是1−|θ|/21−|θ|/2。coding部分用得到。

**16. D**

做出來大約都是0.2917…左右。

**17. B**

做出來都是0.0238…左右。

**18. E**

做出來都是0.366…左右。

**19. C**

做出來都是0.0513…左右。

**20. A**

做出來都是0.00515…左右。

程式碼如下，只要輸入題號即可跑那一題。  
如果使用linux：

for i in $(seq 16 20); do python ML\_HW2.py <<< "${i}"; done

即可一次跑所有題目。  
範例輸出：

$ for i in $(seq 16 20); do python ML\_HW2.py <<< "${i}"; done

16 : 0.29197966919312257

17 : 0.023632776332855543

18 : 0.3683015959104666

19 : 0.05008424263855399

20 : 0.005353162148623501

(程式碼貼在下面, 不過也有直接的檔案：<https://www.csie.ntu.edu.tw/~b08902068/ML_HW/ML_HW2_prog_part.py>)

import numpy as np

import random

import math

def GenData(N, tau):

"""

Generate `N` data points (x, y), with noice probability `tau`

"""

# create number in [0,1), then having probability 1/2 to be positive or negative

# note that np.sign(1) = 1, np.sign(-1) = -1, np.sign(0) = 0

x = [random.random() \* np.sign(random.random() - 0.5) for \_ in range(N)]

# uses sign of x, with probability `tau` to change sign

# note that Pr(np.sign(random.random() - tau) == -1) == tau

y = [np.sign(i) \* np.sign(random.random() - tau) for i in x]

return list(zip(x, y))

def DecitionStumpAlgo(data):

"""

runs decition stump learning algorithm, returning (s, theta, E\_in)

data: a list containing all (x, y), sorted when initialzing

"""

data.sort(key = lambda a: a[0])

# theta\_set: -1 and all means of x\_i and x\_{i+1}

theta\_set = [-1] + [(data[i][0] + data[i+1][0]) / 2 for i in range(len(data) - 1)]

# we define dp[s][t\_i] to be `how many points is predicted correctly by

# h\_{s, theta\_set(t\_i)}`, and we have transition function:

# (note that theta\_set[t\_i] lies between data[t\_i-1] and data[t\_i])

# dp[s][t\_i] =

# (if s == -1) len(data) - dp[-s][t\_i]

# (if s == +1)

# (if t\_i == 0) count how many y\_i == 1

# (else) dp[s][t\_i - 1] - sign(data[t\_i-1][1])

dp\_plus = np.zeros(len(data)) # dp[+1] (as for dp[-1], we can generate later)

for t\_i in range(len(data)):

if t\_i == 0:

# count how many y\_i == 1

dp\_plus[t\_i] = len(list(filter(

lambda a: a[1] > 0,

data

)))

else:

dp\_plus[t\_i] = dp\_plus[t\_i - 1] - np.sign(data[t\_i-1][1])

dp\_minus = len(data) - dp\_plus

# find min(E\_in) by max(#predict correctly)

# returns s, theta, E\_in

plus\_max\_theta\_index = np.argmax(dp\_plus)

minus\_max\_theta\_index = np.argmax(dp\_minus)

if(dp\_plus[plus\_max\_theta\_index] > dp\_minus[minus\_max\_theta\_index]): # s = +1 wins

E\_in = (len(data) - dp\_plus[plus\_max\_theta\_index])/len(data)

return 1, theta\_set[plus\_max\_theta\_index], E\_in

else: # s = -1 wins

E\_in = (len(data) - dp\_minus[minus\_max\_theta\_index])/len(data)

return -1, theta\_set[minus\_max\_theta\_index], E\_in

if \_\_name\_\_ == "\_\_main\_\_":

# storing all problem's parameter

# (since all problem only differs by `tau` and `data\_n`)

problem\_parameter = {

# tau data\_n

"16": ( 0, 2 ),

"17": ( 0, 20 ),

"18": ( 0.1, 2 ),

"19": ( 0.1, 20 ),

"20": ( 0.1, 200 ),

}

problem\_number = input()

tau, data\_n = problem\_parameter[problem\_number]

delta\_ls = [] # storing E\_out - E\_in

for \_ in range(10000):

data = GenData(data\_n, tau)

s, theta, E\_in = DecitionStumpAlgo(data)

# E\_out with tau = 0

E\_out = (abs(theta)/2) if s > 0 else (1 - abs(theta)/2)

# E\_out with tau, by problem 11

E\_out = E\_out + tau - 2\*E\_out\*tau

delta\_ls.append(E\_out-E\_in)

print(problem\_number, ":", np.mean(delta\_ls))

發表於 [**HackMD**](https://hackmd.io/)

 65

[讚賞](https://hackmd.io/@Kaiserouo/BkgXnlPwD) [收藏](https://hackmd.io/@Kaiserouo/BkgXnlPwD) [訂閱](https://hackmd.io/@Kaiserouo/BkgXnlPwD)