**ML Foundation HW3**

**1. B**

直接丟wolfram alpha解0.01⋅(1−12N)≥0.0060.01⋅(1−12N)≥0.006(其實也可手解)，得N≥30N≥30。

**2. A**

**( a )**

是，我們整理：  
XTXw=XTyXT(Xw−y)=0⇒Xw−y∈N(AT)XTXw=XTyXT(Xw−y)=0⇒Xw−y∈N(AT)  
N(AT)N(AT)代表ATAT的nullspace。根據Fundamental theorem of linear algebra，我們知道：  
C(A)=(N(AT))⊥C(A)=(N(AT))⊥  
C(A)C(A)是AA的column space。所以我們知道Xw−yXw−y會垂直於C(A)C(A)。  
我們也知道Xw∈C(A)Xw∈C(A)。所以只要Xw=projC(A)yXw=projC(A)y，那Xw−yXw−y也會垂直於C(x)C(x)。  
所以對於所有yy，存在ww使得normal equation有解。注意因為projC(A)y∈C(A)projC(A)y∈C(A)，所以的確存在一ww，使得Xw=projC(A)yXw=projC(A)y。

**( b )**

不，∇Ein(w)=0∇Ein(w)=0，但是Ein(w)Ein(w)不一定。簡單來說最小值不一定是0。  
除非Xw=yXw=y不然不會Ein(w)=0Ein(w)=0。但是XX不一定是invertible的方陣，所以也沒辦法直接從normal equation倒推Xw=yXw=y.

**( c )**

簡單來說，w=(XTX)−1XTyw=(XTX)−1XTy，但是要Xw=yXw=y，需要X(XTX)−1XT=IX(XTX)−1XT=I，因為XX不一定是方陣所以不一定成立。  
例如X=⎡⎢

⎢

⎢⎣12376479⎤⎥

⎥

⎥⎦X=[12376479]，有X(XTX)−1XT=⎡⎢

⎢

⎢

⎢

⎢

⎢⎣9215743−24323215743277430−582153186−243−5821516421514432321531861443237430⎤⎥

⎥

⎥

⎥

⎥

⎥⎦X(XTX)−1XT=[9215743−24323215743277430−582153186−243−5821516421514432321531861443237430]

**( d )**

Ein(w)=0Ein(w)=0，代表Xw=yXw=y。所以ww至少是normal equation的其中一解。問題是假如dim(Nullspace(X))>0dim(Nullspace(X))>0，又yy剛好落在XX的column space上，代表會有無限多個ww使得Ein(w)=0Ein(w)=0。當然連帶地有無限多個ww符合normal equation。  
例如X=⎡⎢

⎢

⎢⎣12243648⎤⎥

⎥

⎥⎦,y=⎡⎢

⎢

⎢⎣1234⎤⎥

⎥

⎥⎦X=[12243648],y=[1234]，有w=[10]+s[2−1]w=[10]+s[2−1]，符合Xw=yXw=y。

**3. C**

**( a )**

HH不變，因為把input向量變兩倍還是會吐出兩倍後的結果。  
或說(2X)((2X)T(2X))−1(2X)T=44H=H(2X)((2X)T(2X))−1(2X)T=44H=H。或說XX的coluumn space不變。

**( b )**

HH不變，因為HH是把yy投影到XX的column space，但是把某行乘一數並沒有改變column space，所以不變。

**( c )**

HH可能會變，因為XX的column space會變動。  
或隨便舉例，X=⎡⎢

⎢

⎢⎣1428310416⎤⎥

⎥

⎥⎦,H=⎡⎢

⎢

⎢

⎢

⎢⎣12122104212214210821001042182101621⎤⎥

⎥

⎥

⎥

⎥⎦X=[1428310416],H=[12122104212214210821001042182101621]，但是X′=⎡⎢

⎢

⎢

⎢⎣1414110314⎤⎥

⎥

⎥

⎥⎦,H′=⎡⎢

⎢

⎢

⎢

⎢⎣1313013131301300101313013⎤⎥

⎥

⎥

⎥

⎥⎦X′=[1414110314],H′=[1313013131301300101313013]。

**( d )**

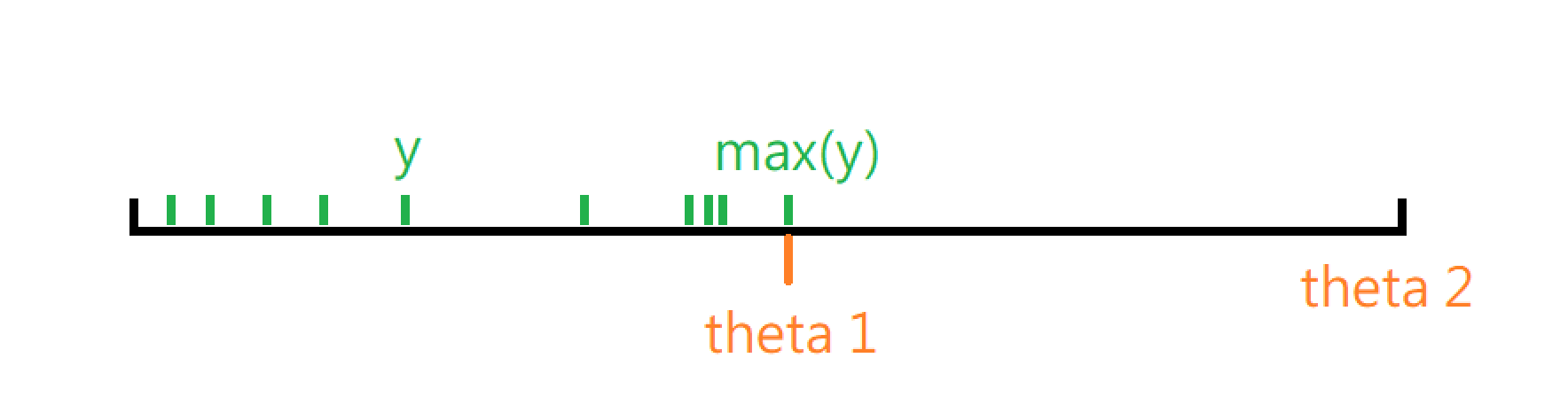
HH不會變，因為XX的column space不變。

**4. E**

* 是，根據Hoeffding’s Inequality。
* 是，首先我們知道likelihood(^θ):=lh(^θ)=∏Ni=1P(xi)P(y|^θ)likelihood(θ^):=lh(θ^)=∏i=1NP(xi)P(y|θ^)。  
  我們知道P(1|^θ)=^θ,P(0|^θ)=1−^θP(1|θ^)=θ^,P(0|θ^)=1−θ^經過處理後，我們知道lh(^θ)lh(θ^)跟∑Ni=1ln(P(yi|^θ)):=f(^θ)∑i=1Nln⁡(P(yi|θ^)):=f(θ^)正相關，我們只要maximize此式即可。  
  我們假設y=1y=1的有aa個，y=0y=0的有bb個，可得：  
  df(^θ)d^θ=d(aln(^θ)+bln(1−^θ))d^θ=a^θ−b1−^θ=0⇒a(1−^θ)−b^θ=0^θ=aa+b=νdf(θ^)dθ^=d(aln⁡(θ^)+bln⁡(1−θ^))dθ^=aθ^−b1−θ^=0⇒a(1−θ^)−bθ^=0θ^=aa+b=ν
* 是，做dEin(^y)/d^y=2^y−2ν=0dEin(y^)/dy^=2y^−2ν=0，可得^y=∑Ni=1yn=νy^=∑i=1Nyn=ν。而因為他不會有最大值，所以這是最小值無誤。
* 是，根據上面微分式。

所以四個，E。

**5. D**

我們用推理的方式。因為這是連續的分布所以除了純推理外我也不知道怎麼做了。  
我們先訂^θ1=max(y1...yN)θ^1=max(y1...yN)和^θ2=2max(y1...yN)θ^2=2max(y1...yN)：  
  
我們可以把likelihood(^θ2):=lh(^θ2)likelihood(θ^2):=lh(θ^2)轉換成兩個procedure：

1. 以uniform distribution抽兩個區間(也就是各50%)：[0,^θ1],[^θ1,^θ2][0,θ^1],[θ^1,θ^2]
2. 在選到的那個區間裡抽點

我們知道選到除了[0,^θ1][0,θ^1]之外的區間都是不可能抽到任何yiyi的；如果幸運抽到這個區間的話，那likelihood即為lh(^θ1)lh(θ^1)。  
所以我們可以靠^θ1θ^1來當作一個基準，藉此觀察其他^θθ^的相對likelihood。  
我們令^θ=k⋅^θ1θ^=k⋅θ^1，可知抽到區間[0,^θ1][0,θ^1]的機率為1k=^θ1^θ=max(y1...yN)^θ1k=θ^1θ^=max(y1...yN)θ^。  
答案d。

**6. B**

首先，我們把式子換成gradient descend的式子：  
wt+1←wt+ηN∑n:yn≠sign(wTtxn)ynxn⇔wt+1←wt−η∇Ein(wt)⇔wt+1←wt−η′N∑n∈[1,N],n∈Nddwterr(wt,xn,yn)wt+1←wt+ηN∑n:yn≠sign(wtTxn)ynxn⇔wt+1←wt−η∇Ein(wt)⇔wt+1←wt−η′N∑n∈[1,N],n∈Nddwterr(wt,xn,yn)  
因為只取預測錯誤的點，所以我們的error function需要符合兩個條件：

* yn≠sign(wTtxn)yn≠sign(wtTxn)，或說ynwTtxn<0ynwtTxn<0時，需要ddwterr(wt,xn,yn)=−ynxnddwterr(wt,xn,yn)=−ynxn。
* yn=sign(wTtxn)yn=sign(wtTxn)，或說ynwTtxn>0ynwtTxn>0時，需要ddwterr(wt,xn,yn)=0ddwterr(wt,xn,yn)=0。

我們取err(w,x,y)=max(0,−ywTx)err(w,x,y)=max(0,−ywTx)，因為可發現它會遮蔽ynwTtxn>0ynwtTxn>0的點，使之的error function為0，自然微分後亦為0。  
或者直接來：

* ynwTtxn<0ynwtTxn<0，ddwterr(wt,xn,yn)=ddwt(−ynwTxn)=−2ynwTxnddwterr(wt,xn,yn)=ddwt(−ynwTxn)=−2ynwTxn。
* ynwTtxn>0ynwtTxn>0，ddwterr(wt,xn,yn)=ddwt0=0ddwterr(wt,xn,yn)=ddwt0=0。

所以除了常數外沒錯，常數另外在η′η′處理即可。

**7. A**

errexp(w,x,y)=exp(−ywTx)⇒−∇errexp(w,x,y)=−∇exp(−ywTx)=−∇exp(−y⋅∑∀iwi⋅xi)errexp(w,x,y)=exp⁡(−ywTx)⇒−∇errexp(w,x,y)=−∇exp⁡(−ywTx)=−∇exp⁡(−y⋅∑∀iwi⋅xi)  
我們也知道ddwiexp(−y⋅∑∀iwi⋅xi)=−yxi⋅exp(−y⋅∑∀iwi⋅xi)ddwiexp⁡(−y⋅∑∀iwi⋅xi)=−yxi⋅exp⁡(−y⋅∑∀iwi⋅xi)。  
所以−∇exp(−y⋅∑∀iwi⋅xi)=−(−yx⋅exp(−ywTx))=yx⋅exp(−ywTx)−∇exp⁡(−y⋅∑∀iwi⋅xi)=−(−yx⋅exp⁡(−ywTx))=yx⋅exp⁡(−ywTx)。

**8. B**

為簡化，我們定義bE(u):=b,AE(u)=AbE(u):=b,AE(u)=A。  
首先，我們化簡式子：  
w←u+v⇒v=w−uw←u+v⇒v=w−u  
代表E(w)≈E(u)+bTv+12vTAv=f(v)E(w)≈E(u)+bTv+12vTAv=f(v)  
注意AE(u)AE(u)是positive definite，也代表他是symmetric的。所以∇v(vTAv)=2Av∇v(vTAv)=2Av。且∇v(bTv)=b∇v(bTv)=b。  
所以為了minimize f(v)=E(u)+bTv+12vTAvf(v)=E(u)+bTv+12vTAv，需要∇vf(v)=0∇vf(v)=0，又：  
∇vf(v)=∇v(E(u)+bTv+12vTAv)=∇v(bTv+12vTAv)=b+Av=0∇vf(v)=∇v(E(u)+bTv+12vTAv)=∇v(bTv+12vTAv)=b+Av=0  
可得v=−A−1b=−(AE(u))−1bE(u)v=−A−1b=−(AE(u))−1bE(u)

**9. B**

Ein(w)=1N∥Xw−y∥2=1N(wTXTXw−2wTXTy+yTy)Ein(w)=1N‖Xw−y‖2=1N(wTXTXw−2wTXTy+yTy)。  
我們定義A:=XTXA:=XTX且他是symmetric的，b:=XTyb:=XTy，c:=yTyc:=yTy所以我們知Ein(w)=1N(wTAw−2wTb+c)Ein(w)=1N(wTAw−2wTb+c)。  
注意AEin(w)=∇2Ein(w)=∇⋅∇TEin(w)AEin(w)=∇2Ein(w)=∇⋅∇TEin(w)，注意中間的點是乘法不是內積。  
所以：  
AEin(w)=∇⋅∇T(1N(wTAw−2wTb+c))=∇⋅(∇(1N(wTATw−2bTw+c)))T=∇⋅(1N(2ATw−2b))T=2N∇⋅(wTA−bT)AEin(w)=∇⋅∇T(1N(wTAw−2wTb+c))=∇⋅(∇(1N(wTATw−2bTw+c)))T=∇⋅(1N(2ATw−2b))T=2N∇⋅(wTA−bT)  
我們現在定義A=[a1a2a3...]A=[a1a2a3...]，其中aiai代表AA的第ii行。所以wTA=[wTa1wTa2wTa3...]wTA=[wTa1wTa2wTa3...]。注意wTaiwTai是一個函數，值域為RR，代表∇∇乘法丟到wTAwTA裡的每一行會變單純的gradient。其實也可以真的乘開簡化來證明就是了，這裡不再做。  
所以：  
AEin(w)=2N∇⋅(wTA−bT)=2N([∇wTa1∇wTa2∇wTa3...]−∇⋅bT)=2N[a1a2a3...]=2N⋅A=2N⋅XTXAEin(w)=2N∇⋅(wTA−bT)=2N([∇wTa1∇wTa2∇wTa3...]−∇⋅bT)=2N[a1a2a3...]=2N⋅A=2N⋅XTX

**10. B**

我們分兩個case做：y=ky=k和y≠ky≠k。  
假設y=ky=k，我們知道：  
∂hk(x)∂Wik=∂(exp(wTkx)exp(wTkx)+∑Kj=1[y≠k]exp(wTjx))∂Wik∂hk(x)∂Wik=∂(exp⁡(wkTx)exp⁡(wkTx)+∑j=1K[y≠k]exp⁡(wjTx))∂Wik  
事實上，∑Kj=1[y≠k]exp(wTjx)∑j=1K[y≠k]exp⁡(wjTx)就是原本hk(x)hk(x)的分母把exp(wTkx)exp⁡(wkTx)抽離的剩下的部分，因為他們對WikWik微分會是0，我們姑且定義∑Kj=1[y≠k]exp(wTjx):=a∑j=1K[y≠k]exp⁡(wjTx):=a，並且∂a∂Wik=0∂a∂Wik=0。我們也知道：  
∂exp(wTkx)∂Wik=∂exp(∑dj=0Wjkxi)∂Wik=xi⋅exp(wTkx)∂exp⁡(wkTx)∂Wik=∂exp⁡(∑j=0dWjkxi)∂Wik=xi⋅exp⁡(wkTx)  
注意xixi就是xx的第ii象限的值。  
因為我們最主要在乎的也只有exp(wTkx)exp⁡(wkTx)，姑且定義exp(wTkx):=eexp⁡(wkTx):=e且∂e∂Wik=xie∂e∂Wik=xie。  
所以我們有：  
∂hk(x)∂Wik=∂(ee+a)∂Wik=xie(e+a)−xie2(e+a)2=xiea(e+a)2∂hk(x)∂Wik=∂(ee+a)∂Wik=xie(e+a)−xie2(e+a)2=xiea(e+a)2  
所以可得：  
∂err(W,x,y)∂Wik=∂(−ln(hk(x)))∂Wik=−∂hk(x)∂Wik1hk(x)=−xiea(e+a)2e+ae=−xiae+a=(hk(x)−1)xi∂err(W,x,y)∂Wik=∂(−ln⁡(hk(x)))∂Wik=−∂hk(x)∂Wik1hk(x)=−xiea(e+a)2e+ae=−xiae+a=(hk(x)−1)xi  
我們現在看y≠ky≠k：  
∂hy(x)∂Wik=∂(exp(wTyx)exp(wTkx)+∑Kj=1[y≠k]exp(wTjx))∂Wik∂hy(x)∂Wik=∂(exp⁡(wyTx)exp⁡(wkTx)+∑j=1K[y≠k]exp⁡(wjTx))∂Wik  
因為∂exp(wTyx)∂Wik=0∂exp⁡(wyTx)∂Wik=0，姑且令exp(wTyx):=bexp⁡(wyTx):=b，可得：  
∂hy(x)∂Wik=∂(be+a)∂Wik=−bxie(e+a)2∂hy(x)∂Wik=∂(be+a)∂Wik=−bxie(e+a)2  
可得：  
∂err(W,x,y)∂Wik=∂(−ln(hy(x)))∂Wik=−∂hy(x)∂Wik1hy(x)=−−bxie(e+a)2e+ab=−xiee+a=hk(x)⋅xi∂err(W,x,y)∂Wik=∂(−ln⁡(hy(x)))∂Wik=−∂hy(x)∂Wik1hy(x)=−−bxie(e+a)2e+ab=−xiee+a=hk(x)⋅xi

**11. E**

也就是要讓linear regression (binary classification)的∇Ein∇Ein為0。  
從上面的MLR，我們可知w∗1,w∗2w1∗,w2∗做出來的∇w1Ein(W)∇w1Ein(W)和∇w2Ein(W)∇w2Ein(W)都會是0。  
我們假設把所有點(x,y)(x,y)分成兩個集合：A,BA,B：所有y=1y=1的點都在AA，其他y=2y=2的點都在BB。  
可得知：  
∇w1Ein(W)=1N⎡⎣∑(x,y)∈A(h1(x)−1)x+∑(x,y)∈Bh1(x)x⎤⎦=0∇w2Ein(W)=1N⎡⎣∑(x,y)∈Ah2(x)x+∑(x,y)∈B(h2(x)−1)x⎤⎦=0∇w1Ein(W)=1N[∑(x,y)∈A(h1(x)−1)x+∑(x,y)∈Bh1(x)x]=0∇w2Ein(W)=1N[∑(x,y)∈Ah2(x)x+∑(x,y)∈B(h2(x)−1)x]=0  
不過注意我們有兩點性質：

* h1(x)=1−h2(x)h1(x)=1−h2(x)
* h1(x)=exp(wT1x)exp(wT1x)+exp(wT2x)=θ((w1−w2)Tx)h1(x)=exp⁡(w1Tx)exp⁡(w1Tx)+exp⁡(w2Tx)=θ((w1−w2)Tx)，且θθ是sigmoid function。對h2(x)=θ((w2−w1)Tx)h2(x)=θ((w2−w1)Tx)同理。

我們需要使logistic regression的∇wEin(w)=0∇wEin(w)=0，我們有：  
∇wEin(w)=1NN∑i=1θ(−yiwTxi)(−yixi)=0∇wEin(w)=1N∑i=1Nθ(−yiwTxi)(−yixi)=0  
又我們知道：

* (x,y)∈A⇒θ(−ywTx)(−yx)=θ(wTx)x(x,y)∈A⇒θ(−ywTx)(−yx)=θ(wTx)x
* (x,y)∈B⇒θ(−ywTx)(−yx)=θ(−wTx)(−x)(x,y)∈B⇒θ(−ywTx)(−yx)=θ(−wTx)(−x)

所以：  
∇wEin(w)=1NN∑i=1θ(−yiwTxi)(−yixi)=1N⎡⎣∑(x,y)∈Aθ(wTx)x+∑(x,y)∈Bθ(−wTx)(−x)⎤⎦=1N⎡⎣∑(x,y)∈Aθ(wTx)x+∑(x,y)∈B(θ(wTx)−1)x⎤⎦=0∇wEin(w)=1N∑i=1Nθ(−yiwTxi)(−yixi)=1N[∑(x,y)∈Aθ(wTx)x+∑(x,y)∈Bθ(−wTx)(−x)]=1N[∑(x,y)∈Aθ(wTx)x+∑(x,y)∈B(θ(wTx)−1)x]=0  
我們可以發現w=w∗2−w∗1w=w2∗−w1∗的話，θ(wTx)=h2(x)θ(wTx)=h2(x)，所以∇wEin(w)=0∇wEin(w)=0。

**12. E**

如果需要的話，下面的程式只要輸入六個參數就會告訴你這個選項正不正確：

import numpy as np

data = [

[[0,1], -1],

[[1,-0.5], -1],

[[-1,0], -1],

[[-1,2], 1],

[[2,0], 1],

[[0,-2], 1],

]

def sgn(x):

return 1 if x > 0 else -1

def func(para, x):

return np.dot(np.array(para), np.array([

1, x[0], x[1], x[0]\*\*2, x[0]\*x[1], x[1]\*\*2

]))

while True:

para = list(map(lambda x: int(x), input().split()))

a = np.array([sgn(func(para, d[0])) == d[1] for d in data])

print(np.all(a))

**13. B**

雖然這個方法很爛但我只能想到這樣…。  
其實這就是HW2的第2題的generalized版本，從那一題我們應該可以得出generating function：  
G(n,d)=(2n−2)d+2=2nd−2d+2G(n,d)=(2n−2)d+2=2nd−2d+2  
因為我們單看每個維度，最多都可以做出2nd2nd種可能，但是因為全被判斷為1或-1是每個維度都有的，只能算一次，才有上面的式子。  
我們知道對於dVCdVC，應該符合下面條件：  
G(dVC,d)≥2dVCG(dVC+1,d)<2dVC+1G(dVC,d)≥2dVCG(dVC+1,d)<2dVC+1  
第一個式子其實有顯然解dVC≥1dVC≥1，雖然兩個應該都可以做得出來不過我們考慮下面的。  
我們現在直接帶入B選項證明B選項是對的，至少複雜度是對的。我沒辦法從頭開始證所以只能這樣了…。  
我們代入dVC=2(log2d+1)dVC=2(log2⁡d+1)：  
G(dVC+1,d)<2dVC+12(dVC+1)d−2d+1<2dVC+12(2(log2d+1)+1)d−2d+1<22(log2d+1)+14log2d⋅d+6d−2d+1<22log2d23=8d2G(dVC+1,d)<2dVC+12(dVC+1)d−2d+1<2dVC+12(2(log2⁡d+1)+1)d−2d+1<22(log2⁡d+1)+14log2⁡d⋅d+6d−2d+1<22log2⁡d23=8d2  
又我們有d≥4⇒log2d≤d2d≥4⇒log2⁡d≤d2，所以：  
4log2d⋅d+6d−2d+1<22log2d23=8d22d2+4d+1<8d24log2⁡d⋅d+6d−2d+1<22log2⁡d23=8d22d2+4d+1<8d2  
對於d≥4d≥4的確成立。事實上可求d>2+√102d>2+102時即成立。  
也可發現式子左右的複雜度符合，所以應該是沒辦法再壓多少了。此為tight upper bound。或者說從第一條式子做應該會得到類似的式子，導致做出了類似big-theta的效果。

**14. D**

做出來是0.60532238

import numpy as np

import numpy.linalg as la

import random

import math

FILENAME = "hw3\_train.dat.txt" # input file.

X\_0 = 1

MUL\_X = 1

def ReadData(fname):

"""

Read data from file. returns array

Only split lines of file and store in array.

Postprocess into x & y in "DecorateData".

"""

arr = []

with open(fname, "r") as fin:

while (s := fin.readline()) != "":

arr.append(np.array(list(map(lambda x: float(x), s.split()))))

return np.array(arr)

def DecorateData(arr):

"""

Use input array to make D={(x\_n, y\_n)}.

Will add x\_0=X\_0 in front of every x, increasing x's dimension by 1.

all other x\_i will be multiplied by MUL\_X.

x's dimension should be greater than 1 or else format error

"""

x = arr[:, :-1]

y = arr[:, -1]

x = np.hstack((np.ones((x.shape[0], 1))\*X\_0, x\*MUL\_X))

return x, y

def FindPinvW(x, y):

return np.dot(la.pinv(x), y)

def FindSqrEin(w, x, y):

return np.mean([

np.sum((np.dot(w, x[i]) - y[i])\*\*2)

for i in range(len(x))

])

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

x, y = DecorateData(ReadData(FILENAME))

w = FindPinvW(x, y)

print(FindSqrEin(w, x, y))

**15. C**

做出來是1810.131

import numpy as np

import numpy.linalg as la

import random

import math

FILENAME = "hw3\_train.dat.txt" # input file.

EXP\_N = 1000

X\_0 = 1

MUL\_X = 1

ETA = 0.001

def ReadData(fname):

"""

Read data from file. returns array

Only split lines of file and store in array.

Postprocess into x & y in "DecorateData".

"""

arr = []

with open(fname, "r") as fin:

while (s := fin.readline()) != "":

arr.append(np.array(list(map(lambda x: float(x), s.split()))))

return np.array(arr)

def DecorateData(arr):

"""

Use input array to make D={(x\_n, y\_n)}.

Will add x\_0=X\_0 in front of every x, increasing x's dimension by 1.

all other x\_i will be multiplied by MUL\_X.

x's dimension should be greater than 1 or else format error

"""

x = arr[:, :-1]

y = arr[:, -1]

x = np.hstack((np.ones((x.shape[0], 1))\*X\_0, x\*MUL\_X))

return x, y

def FindPinvW(x, y):

return np.dot(la.pinv(x), y)

def FindSqrEin(w, x, y):

return np.mean((np.dot(x, w) - y)\*\*2)

def SGD(x, y, w\_lin, exp\_n=EXP\_N):

def S\_apply(w, x, y):

return w + ETA \* 2\*(y - np.dot(w, x)) \* x

data\_n = x.shape[0] # size of data set

u\_ls = [] # update times list

target\_Ein = 1.01\*FindSqrEin(w\_lin, x, y)

for e\_i in range(exp\_n):

random.seed()

w\_t = np.zeros(x.shape[1])

update\_cnt = 0

while FindSqrEin(w\_t, x, y) > target\_Ein:

# pick random number

r\_i = random.randint(0, data\_n-1)

# apply change for w

update\_cnt += 1

w\_t = S\_apply(w\_t, x[r\_i], y[r\_i])

u\_ls.append(update\_cnt)

# print this experiment's update count

# print("{}:".format(e\_i+1), update\_cnt, flush=True)

return u\_ls

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

x, y = DecorateData(ReadData(FILENAME))

w\_lin = FindPinvW(x, y)

print("Final result:", np.mean(SGD(x, y, w\_lin)), flush=True)

**16. C**

做出來是0.56898679

import numpy as np

import numpy.linalg as la

import random

import math

FILENAME = "hw3\_train.dat.txt" # input file.

EXP\_N = 1000

X\_0 = 1

MUL\_X = 1

ETA = 0.001

def ReadData(fname):

"""

Read data from file. returns array

Only split lines of file and store in array.

Postprocess into x & y in "DecorateData".

"""

arr = []

with open(fname, "r") as fin:

while (s := fin.readline()) != "":

arr.append(np.array(list(map(lambda x: float(x), s.split()))))

return np.array(arr)

def DecorateData(arr):

"""

Use input array to make D={(x\_n, y\_n)}.

Will add x\_0=X\_0 in front of every x, increasing x's dimension by 1.

all other x\_i will be multiplied by MUL\_X.

x's dimension should be greater than 1 or else format error

"""

x = arr[:, :-1]

y = arr[:, -1]

x = np.hstack((np.ones((x.shape[0], 1))\*X\_0, x\*MUL\_X))

return x, y

def FindCEErr(w, x, y):

return np.mean(np.log(1 + np.exp(-y \* (np.dot(x, w)))))

def SGD(x, y, exp\_n=EXP\_N):

def sigmoid(a):

return 1 / (1 + np.exp(-a))

def S\_apply(w, x, y):

return w + ETA \* sigmoid(-y \* np.dot(w, x)) \* (y \* x)

data\_n = x.shape[0] # size of data set

e\_ls = [] # cross entropy error list

for e\_i in range(exp\_n):

random.seed()

w\_t = np.zeros(x.shape[1])

for i in range(500):

# pick random number

r\_i = random.randint(0, data\_n-1)

# apply change for w

w\_t = S\_apply(w\_t, x[r\_i], y[r\_i])

e\_ls.append(FindCEErr(w\_t, x, y))

# print this experiment's error

# print("{}: {}".format(e\_i+1, e\_ls[-1]), flush=True)

return e\_ls

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

x, y = DecorateData(ReadData(FILENAME))

print("Final result:", np.mean(SGD(x, y)), flush=True)

**17. B**

做出來是0.502810448802

import numpy as np

import numpy.linalg as la

import random

import math

FILENAME = "hw3\_train.dat.txt" # input file.

EXP\_N = 1000

X\_0 = 1

MUL\_X = 1

ETA = 0.001

def ReadData(fname):

"""

Read data from file. returns array

Only split lines of file and store in array.

Postprocess into x & y in "DecorateData".

"""

arr = []

with open(fname, "r") as fin:

while (s := fin.readline()) != "":

arr.append(np.array(list(map(lambda x: float(x), s.split()))))

return np.array(arr)

def DecorateData(arr):

"""

Use input array to make D={(x\_n, y\_n)}.

Will add x\_0=X\_0 in front of every x, increasing x's dimension by 1.

all other x\_i will be multiplied by MUL\_X.

x's dimension should be greater than 1 or else format error

"""

x = arr[:, :-1]

y = arr[:, -1]

x = np.hstack((np.ones((x.shape[0], 1))\*X\_0, x\*MUL\_X))

return x, y

def FindPinvW(x, y):

return np.dot(la.pinv(x), y)

def FindCEErr(w, x, y):

return np.mean(np.log(1 + np.exp(-y \* (np.dot(x, w)))))

def SGD(x, y, w\_lin, exp\_n=EXP\_N):

def sigmoid(a):

return 1 / (1 + np.exp(-a))

def S\_apply(w, x, y):

return w + ETA \* sigmoid(-y \* np.dot(w, x)) \* (y \* x)

data\_n = x.shape[0] # size of data set

e\_ls = [] # cross entropy error list

for e\_i in range(exp\_n):

random.seed()

w\_t = w\_lin.copy()

for i in range(500):

# pick random number

r\_i = random.randint(0, data\_n-1)

# apply change for w

w\_t = S\_apply(w\_t, x[r\_i], y[r\_i])

e\_ls.append(FindCEErr(w\_t, x, y))

# print this experiment's error

# print("{}: {}".format(e\_i+1, e\_ls[-1]), flush=True)

return e\_ls

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

x, y = DecorateData(ReadData(FILENAME))

w\_lin = FindPinvW(x, y)

print("Final result:", np.mean(SGD(x, y, w\_lin)), flush=True)

**18. A**

做出來是0.3226666666666

import numpy as np

import numpy.linalg as la

import random

import math

FILENAME\_TRAIN = "hw3\_train.dat.txt" # input file.

FILENAME\_TEST = "hw3\_test.dat.txt" # input file.

X\_0 = 1

MUL\_X = 1

def ReadData(fname):

"""

Read data from file. returns array

Only split lines of file and store in array.

Postprocess into x & y in "DecorateData".

"""

arr = []

with open(fname, "r") as fin:

while (s := fin.readline()) != "":

arr.append(np.array(list(map(lambda x: float(x), s.split()))))

return np.array(arr)

def DecorateData(arr):

"""

Use input array to make D={(x\_n, y\_n)}.

Will add x\_0=X\_0 in front of every x, increasing x's dimension by 1.

all other x\_i will be multiplied by MUL\_X.

x's dimension should be greater than 1 or else format error

"""

x = arr[:, :-1]

y = arr[:, -1]

x = np.hstack((np.ones((x.shape[0], 1))\*X\_0, x\*MUL\_X))

return x, y

def FindPinvW(x, y):

return np.dot(la.pinv(x), y)

def Find01error(w, x, y):

return np.mean(np.dot(x, w) \* y < 0)

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

x\_train, y\_train = DecorateData(ReadData(FILENAME\_TRAIN))

x\_test, y\_test = DecorateData(ReadData(FILENAME\_TEST))

w\_lin = FindPinvW(x\_train, y\_train)

print("Final result:", abs(

Find01error(w\_lin, x\_train, y\_train) - Find01error(w\_lin, x\_test, y\_test)

), flush=True)

**19. B**

做出來是0.373666666666

import numpy as np

import numpy.linalg as la

import random

import math

FILENAME\_TRAIN = "hw3\_train.dat.txt" # input file.

FILENAME\_TEST = "hw3\_test.dat.txt" # input file.

X\_0 = 1

MUL\_X = 1

def ReadData(fname):

"""

Read data from file. returns array

Only split lines of file and store in array.

Postprocess into x & y in "DecorateData".

"""

arr = []

with open(fname, "r") as fin:

while (s := fin.readline()) != "":

arr.append(np.array(list(map(lambda x: float(x), s.split()))))

return np.array(arr)

def DecorateData(arr):

"""

Use input array to make D={(x\_n, y\_n)}.

Will add x\_0=X\_0 in front of every x, increasing x's dimension by 1.

all other x\_i will be multiplied by MUL\_X.

x's dimension should be greater than 1 or else format error

"""

x = arr[:, :-1]

y = arr[:, -1]

x = np.hstack((np.ones((x.shape[0], 1))\*X\_0, x\*MUL\_X))

return x, y

def Transform(x, Q):

# note that x\_0 is included! need to take that out to form pure\_x

pure\_x = x[:, 1:]

transform\_x = x.copy()

for q in range(2, Q+1):

transform\_x = np.hstack((

transform\_x, pure\_x\*\*q

))

return transform\_x

def FindPinvW(x, y):

return np.dot(la.pinv(x), y)

def Find01error(w, x, y):

return np.mean(np.dot(x, w) \* y < 0)

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

x\_train, y\_train = DecorateData(ReadData(FILENAME\_TRAIN))

x\_test, y\_test = DecorateData(ReadData(FILENAME\_TEST))

x\_train = Transform(x\_train, 3)

x\_test = Transform(x\_test, 3)

w\_lin = FindPinvW(x\_train, y\_train)

print("Final result:", abs(

Find01error(w\_lin, x\_train, y\_train) - Find01error(w\_lin, x\_test, y\_test)

), flush=True)

**20. D**

做出來是0.44666666666666  
code就是第19題把main的Transform參數Q改掉而已

import numpy as np

import numpy.linalg as la

import random

import math

FILENAME\_TRAIN = "hw3\_train.dat.txt" # input file.

FILENAME\_TEST = "hw3\_test.dat.txt" # input file.

X\_0 = 1

MUL\_X = 1

def ReadData(fname):

"""

Read data from file. returns array

Only split lines of file and store in array.

Postprocess into x & y in "DecorateData".

"""

arr = []

with open(fname, "r") as fin:

while (s := fin.readline()) != "":

arr.append(np.array(list(map(lambda x: float(x), s.split()))))

return np.array(arr)

def DecorateData(arr):

"""

Use input array to make D={(x\_n, y\_n)}.

Will add x\_0=X\_0 in front of every x, increasing x's dimension by 1.

all other x\_i will be multiplied by MUL\_X.

x's dimension should be greater than 1 or else format error

"""

x = arr[:, :-1]

y = arr[:, -1]

x = np.hstack((np.ones((x.shape[0], 1))\*X\_0, x\*MUL\_X))

return x, y

def Transform(x, Q):

# note that x\_0 is included! need to take that out to form pure\_x

pure\_x = x[:, 1:]

transform\_x = x.copy()

for q in range(2, Q+1):

transform\_x = np.hstack((

transform\_x, pure\_x\*\*q

))

return transform\_x

def FindPinvW(x, y):

return np.dot(la.pinv(x), y)

def Find01error(w, x, y):

return np.mean(np.dot(x, w) \* y < 0)

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

x\_train, y\_train = DecorateData(ReadData(FILENAME\_TRAIN))

x\_test, y\_test = DecorateData(ReadData(FILENAME\_TEST))

x\_train = Transform(x\_train, 10)

x\_test = Transform(x\_test, 10)

w\_lin = FindPinvW(x\_train, y\_train)

print("Final result:", abs(

Find01error(w\_lin, x\_train, y\_train) - Find01error(w\_lin, x\_test, y\_test)

), flush=True)

發表於 [**HackMD**](https://hackmd.io/)

 42

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