**ML Foundation HW4**

**1. C**

我們用線性代數的廣義向量空間的性質得出h(x)h(x)。  
我們定義內積空間C([0,2])C([0,2])是在[0,2][0,2]區間連續的函數集合，我們定義向量內積函數⟨,⟩⟨,⟩：  
f(x),g(x)∈C([0,2])⇒⟨f(x),g(x)⟩:=∫20f(x)⋅g(x)dxf(x),g(x)∈C([0,2])⇒⟨f(x),g(x)⟩:=∫02f(x)⋅g(x)dx  
所以我們有∥f(x)−g(x)∥2=⟨f(x)−g(x),f(x)−g(x)⟩=∫20(f(x)−g(x))2dx‖f(x)−g(x)‖2=⟨f(x)−g(x),f(x)−g(x)⟩=∫02(f(x)−g(x))2dx，可藉由投影得出以最小平方法得出的函數。  
我們把函數exex投影到基底B={x}B={x}上去，可得：  
projBex=projxex=⟨ex,x⟩∥x∥2x=∫20x⋅exdx∫20x2dxx=e2+18/3x=3e2+38xprojBex=projxex=⟨ex,x⟩‖x‖2x=∫02x⋅exdx∫02x2dxx=e2+18/3x=3e2+38x  
所以deterministic noice即為跟原函數的差，|ex−3e2+38x||ex−3e2+38x|。

**2. B**

我們可得知ED(Ein(A(D)))=ED(Eout(A(D)))ED(Ein(A(D)))=ED(Eout(A(D)))是有可能的，考慮noiceless的decision stump，給定P(x)P(x)圍在[−1,1][−1,1]的uniform distribution且y=sign(x)y=sign(x)，與A(D)=h(x)=sign(x)A(D)=h(x)=sign(x)，我們知道A(D)A(D)的確會產出最小EinEin的hh，且我們也知道A(D)=h∗A(D)=h∗，所以成立。  
ED(Ein(A(D)))<ED(Eout(A(D)))ED(Ein(A(D)))<ED(Eout(A(D)))也是有可能的，基本上投影片上面就是個例子了。  
因為ED(Ein(A(D)))>ED(Eout(A(D)))ED(Ein(A(D)))>ED(Eout(A(D)))幾乎不可能：我們的A(D)A(D)會讓Ein(h)Ein(h)最小，所以選B…。

**3. D**

我們舉簡單一點的例子，令d+1=2,N=1d+1=2,N=1，所以：  
Xh=⎡⎢⎣||x1~x1||⎤⎥⎦=[x11~x11x12~x12]Xh=[||x1x1~||]=[x11x11~x12x12~]  
注意我們定義xij=(xi)jxij=(xi)j我們有：  
XThXh=[x11x11+~x11~x11x12x11+~x12~x11x12x11+~x12~x11x22x22+~x22~x22]XhTXh=[x11x11+x11~x11~x12x11+x12~x11~x12x11+x12~x11~x22x22+x22~x22~]  
注意因為每個~xx~都有一個Gaussian noice的ϵϵ項，我們現在先假設a,ba,b兩個變數有同樣的noice ϵϵ，我們知道：  
(a+ϵ)(b+ϵ)=ab+aϵ+bϵ+ϵϵ(a+ϵ)(b+ϵ)=ab+aϵ+bϵ+ϵϵ  
因為我們的Gaussian noice有平均0與變異數σ2σ2，我們知道aϵ,bϵaϵ,bϵ都有期望值0，且ϵϵϵϵ在iid的狀況下有期望值E(ϵ2)=σ2E(ϵ2)=σ2。注意變異數的定義是σ2=E((X−μ)2)=E(X2)σ2=E((X−μ)2)=E(X2)。  
如果上面的兩個ϵϵ互相獨立，那E(ϵaϵb)=E(ϵa)E(ϵb)=0E(ϵaϵb)=E(ϵa)E(ϵb)=0。所以E((a+ϵa)(b+ϵb))=abE((a+ϵa)(b+ϵb))=ab。  
上面舉例的XhXh中，都可以套用上面的a,ba,b的式子：

* 如果a,ba,b有一模一樣的ϵϵ，E((a+ϵ)(b+ϵ))=ab+σ2E((a+ϵ)(b+ϵ))=ab+σ2。
* 如果a,ba,b的ϵϵ不一樣(獨立)但是都是由題目的distribution產生，E((a+ϵa)(b+ϵb))=abE((a+ϵa)(b+ϵb))=ab。

所以上面舉例的XhXh會變成：  
E(XThXh)=[x11x11+~x11~x11x12x11+~x12~x11x12x11+~x12~x11x22x22+~x22~x22]=[x11x11+x11x11+σ2x12x11+x12x11x12x11+x12x11x22x22+x22x22+σ2]=2XTX+Nσ2⋅Id+1E(XhTXh)=[x11x11+x11~x11~x12x11+x12~x11~x12x11+x12~x11~x22x22+x22~x22~]=[x11x11+x11x11+σ2x12x11+x12x11x12x11+x12x11x22x22+x22x22+σ2]=2XTX+Nσ2⋅Id+1  
其實generalize後也是一樣：我們定義~xk=xN+kxk~=xN+k，有：  
(XThXh)ij=2N∑k=1xkixkj(XhTXh)ij=∑k=12Nxkixkj  
但是E(xN+kixN+kj)=xkixkj+σ2⋅[[i=j]]E(xN+kixN+kj)=xkixkj+σ2⋅[[i=j]]，所以最後對角線的每項會多出NN個σ2σ2，且每個xkixkjxkixkj被重複加一次，所以乘二，變成2XTX+Nσ2Id+12XTX+Nσ2Id+1。

**4. E**

我們知道：  
XThyh=⎡⎢⎣||x1...~xn||⎤⎥⎦[yy]=[x11y1+...+~x11y1+......]XhTyh=[||x1...xn~||][yy]=[x11y1+...+x11~y1+......]  
簡單來說，xiyixiyi都會重複一遍，並且把~xijxij~拆成xij+ϵixij+ϵi後會發現XThyh=2XTy+ϵTyXhTyh=2XTy+ϵTy，且ϵ=[ϵ1...ϵN]Tϵ=[ϵ1...ϵN]T。  
我們知道E(ϵTy)=0E(ϵTy)=0，所以E(XThyh)=2XTyE(XhTyh)=2XTy。

**5. D**

wlin=(ZTZ)−1ZTy=(QTXTXQ)−1ZTy=(QTQΓQTQ)−1ZTy=Γ−1ZTywreg=(ZTZ+λI)−1ZTy=(QTQΓQTQ+λI)−1ZTy=(Γ+λI)−1ZTywlin=(ZTZ)−1ZTy=(QTXTXQ)−1ZTy=(QTQΓQTQ)−1ZTy=Γ−1ZTywreg=(ZTZ+λI)−1ZTy=(QTQΓQTQ+λI)−1ZTy=(Γ+λI)−1ZTy  
注意Γ−1Γ−1和(Γ+λI)−1(Γ+λI)−1都是diagonal，代表他們只會逐列乘某一常數。  
我們知道(Γ−1)ii=1γi(Γ−1)ii=1γi且((Γ+λI)−1)ii=1γi+λ((Γ+λI)−1)ii=1γi+λ，所以uivi=1/(γi+λ)1/γi=γiγi+λuivi=1/(γi+λ)1/γi=γiγi+λ。

**6. A**

事實上，我們直接帶入公式來解w∗w∗即可：  
X=⎡⎢

⎢⎣x1⋮xN⎤⎥

⎥⎦,y=⎡⎢

⎢⎣y1⋮yN⎤⎥

⎥⎦⇒w∗=(XTX+λI)−1XTy=(N∑n=1x2n+λ)−1⋅N∑m=1xmym=∑Nm=1xmym∑Nn=1x2n+λX=[x1⋮xN],y=[y1⋮yN]⇒w∗=(XTX+λI)−1XTy=(∑n=1Nxn2+λ)−1⋅∑m=1Nxmym=∑m=1Nxmym∑n=1Nxn2+λ  
既然C=(w∗)2C=(w∗)2，選項為A。

**7. D**

令1N∑Nn=1(y−yn)2+2KNΩ(y):=f(y)1N∑n=1N(y−yn)2+2KNΩ(y):=f(y)，我們知道p∗:=∑Nn=1yn+KN+2Kp∗:=∑n=1Nyn+KN+2K是miny∈Rf(y)miny∈Rf(y)的最佳解，代表f′(p∗)=0f′(p∗)=0。

f′(y)=1NN∑n=1(2(y−yn))+2KNΩ′(y)⇒f′(p∗)=2p∗−2∑Nn=1ynN+2KNΩ′(p∗)f′(y)=1N∑n=1N(2(y−yn))+2KNΩ′(y)⇒f′(p∗)=2p∗−2∑n=1NynN+2KNΩ′(p∗)  
我們定義Ω(y)=(y+α)2⇒Ω′(y)=2(y+α)Ω(y)=(y+α)2⇒Ω′(y)=2(y+α)  
f′(p∗)=2p∗−2∑Nn=1ynN+2KNΩ′(p∗)⇒Ω′(p∗)=2(p∗+α)=N2K(2∑Nn=1ynN−2p∗)=∑Nn=1yn−Np∗K⇒2α=∑Nn=1ynK−NKp∗−2p∗=∑Nn=1ynK−N+2KKp∗=∑Nn=1ynK−(∑Nn=1ynK+1)=−1⇒α=−0.5f′(p∗)=2p∗−2∑n=1NynN+2KNΩ′(p∗)⇒Ω′(p∗)=2(p∗+α)=N2K(2∑n=1NynN−2p∗)=∑n=1Nyn−Np∗K⇒2α=∑n=1NynK−NKp∗−2p∗=∑n=1NynK−N+2KKp∗=∑n=1NynK−(∑n=1NynK+1)=−1⇒α=−0.5  
所以Ω(y)=(y−0.5)2Ω(y)=(y−0.5)2。

**8. B**

首先，因為對於diagonal matrix DD，有∇w(wTDw)=2Dw∇w(wTDw)=2Dw，我們假設Ω(w)=wTDwΩ(w)=wTDw且∇wΩ(w)=2Dw∇wΩ(w)=2Dw。  
參照講義的證明，我們有：  
∇Ein(w)+λN∇Ω(w)=0⇒w=(XTX+λD)−1XTy∇Ein(w)+λN∇Ω(w)=0⇒w=(XTX+λD)−1XTy  
再看~ww~的式子，因為Φ(x)=Γ−1xΦ(x)=Γ−1x，所以定義Z=XΓ−1Z=XΓ−1，且Z,XZ,X的定義照課堂上的定法。  
我們知道：  
~w=(ZTZ+λI)−1ZTy=((Γ−1)TXTXΓ−1+λI)−1(Γ−1)TXTy=(Γ−1XTXΓ−1+λI)−1Γ−1XTy=Γ(XTX+λΓ2)−1XTyw~=(ZTZ+λI)−1ZTy=((Γ−1)TXTXΓ−1+λI)−1(Γ−1)TXTy=(Γ−1XTXΓ−1+λI)−1Γ−1XTy=Γ(XTX+λΓ2)−1XTy  
所以我們知道給定D=Γ2D=Γ2，有~wΓ−1=ww~Γ−1=w，不過這就是我們要的：一個xx用ΦΦ轉換後用~ww~轉換跟直接用ww轉換應該是一樣的，意即~wTΦ(x)=wTxw~TΦ(x)=wTx。  
所以Ω(w)=wTΓ2wΩ(w)=wTΓ2w。

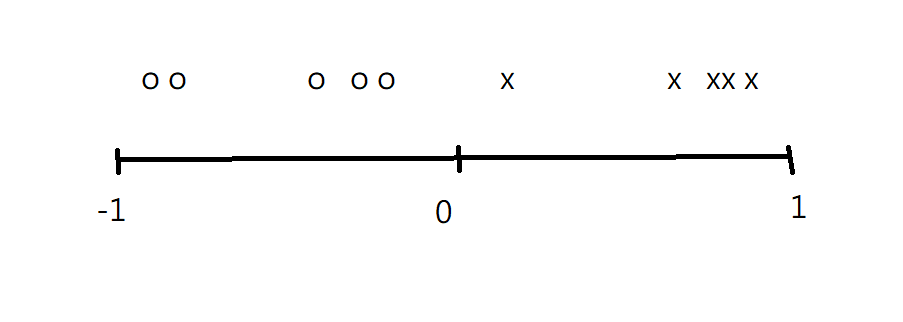
**9. B**

首先，整個EaugEaug的scaling應該是不會影響結果的，所以我們可以安全無視上式的1/N1/N與下式的1/(N+K)1/(N+K)。所以我們只要保證：  
λd∑i=1βiw2i=K∑k=1(wT~xk−~yk)2λ∑i=1dβiwi2=∑k=1K(wTx~k−y~k)2  
首先，參考左式可發現y=0y=0，且我們需要製造一個λλ和βiβi，所以X=√λ⋅√BX=λ⋅B，注意√BB是pointwise square root。

**10. E**

假設我們把其中一筆positive當作leave one out的data，可知剩下的會是negative較多，導致AA會選擇判斷成negative，所以必定為錯，err=1err=1。取negative也一樣。  
EloocvEloocv為所有取的可能性的平均，為1。

**11. C**

我們看圖：  
  
我們知道τ=0τ=0，所以他們不會有錯邊的情形。  
如果我們挑「不是最靠近0的o, x的點」的話，我們知道他做出來的結果必定會分對(因為最靠近0的o, x的點會分對，所以o左邊、x右邊的點也會分對)。所以err=0err=0。  
如果挑最靠近0的o, x的點，WLOG，挑x的話，hh的θθ會需要用那個o與第二靠近0的x做平均。不能保證那個x會被分對，最高err=1err=1。  
所以NN個點中有兩個點可能分錯，max(Eloocv)=2/Nmax(Eloocv)=2/N。

**12. E**

對於constant，我們知道：  
Eloocv=13⎡⎢⎣4h(x)=0+1h(x)=1+1h(x)=1⎤⎥⎦=63Eloocv=13[4⏟h(x)=0+1⏟h(x)=1+1⏟h(x)=1]=63  
對linear，我們也可以求出：

* (−3,0),(3,0)→y=0,err=4(−3,0),(3,0)→y=0,err=4
* (−3,0),(p,2)→y=2x+6p+3,err=(12p+3)2(−3,0),(p,2)→y=2x+6p+3,err=(12p+3)2
* (3,0),(p,2)→y=2x−6p−3,err=(12p−3)2(3,0),(p,2)→y=2x−6p−3,err=(12p−3)2

所以：  
Eloocv=13[4+(12p+3)2+(12p−3)2]=63Eloocv=13[4+(12p+3)2+(12p−3)2]=63  
所以我們有(12p+3)2+(12p−3)2=2(12p+3)2+(12p−3)2=2，求解可得p=√81+36√6p=81+366。

**13. D**

Eval=1K∑Ki=1err(h(xi,yi))Eval=1K∑i=1Kerr(h(xi,yi))  
不過我們知道每個變數(xi,yi),(xj,yj),i≠j(xi,yi),(xj,yj),i≠j是iid的，代表他們之間的covariance應為0，代表：  
Var(Eval)=Var(1KK∑i=1err(h(xi,yi)))=1K2K∑i=1Var(err(h(xi,yi)))=KK2Var(err(h(x),y))Var(Eval)=Var(1K∑i=1Kerr(h(xi,yi)))=1K2∑i=1KVar(err(h(xi,yi)))=KK2Var(err(h(x),y))  
所以答案為1/K1/K。

**14. C**

在所有可能中，只有：

xo ox

ox xo

這兩種無法分開，minEin=1/4minEin=1/4。  
所以E(minEin)=2⋅1/416=2/64E(minEin)=2⋅1/416=2/64

**15. A**

我們知道：  
Eout(g)=P(y=+1)P(g(x)=−1|y=+1)+P(y=−1)P(g(x)=+1|y=−1)Eout(g)=P(y=+1)P(g(x)=−1|y=+1)+P(y=−1)P(g(x)=+1|y=−1)  
所以給定P(y=+1)=pP(y=+1)=p，可知：  
Eout(g)=P(y=+1)P(g(x)=−1|y=+1)+P(y=−1)P(g(x)=+1|y=−1)=pϵ++(1−p)ϵ−=Eout(gc)=1−pEout(g)=P(y=+1)P(g(x)=−1|y=+1)+P(y=−1)P(g(x)=+1|y=−1)=pϵ++(1−p)ϵ−=Eout(gc)=1−p  
解出pp，有p=1−ϵ−1+ϵ+−ϵ−p=1−ϵ−1+ϵ+−ϵ−。

**16. B**

他們的-s 0跟我們欲求的函式有些不同之處：

* 我們要先做xΦ2→zx→Φ2z
* 因為他們的函式是12wTw+C∑log(1+exp(−yiwTx))12wTw+C∑log⁡(1+exp⁡(−yiwTx))，我們的是λNwTw+1N∑log(1+exp(−yiwTx))λNwTw+1N∑log⁡(1+exp⁡(−yiwTx))，我們知道常數倍的saling不影響最後解，所以只需要保持12:C=λN:1N12:C=λN:1N的關係即可。可知2Cλ=1,C=12λ2Cλ=1,C=12λ。

下面的程式最後跑出[0.13333333333333333, 0.13, 0.19333333333333333, 0.25666666666666665, 0.48333333333333334]，可發現log10λ=−2log10⁡λ=−2有最低Eout(wλ)Eout(wλ)

from liblinearutil import \*

import numpy as np

import numpy.linalg as la

import random

import math

FILENAME\_TRAIN = "../../hw4\_train.dat.txt" # input file.

FILENAME\_TEST = "../../hw4\_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):

"""

Feature transform x into \Phi(x) = [1, x\_i, x\_ix\_j], for all i != j

"""

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

pure\_x = x[:, 1:]

Q = pure\_x.shape[1]

transform\_x = x.copy()

for q in range(0, Q):

tmp\_x = pure\_x[:, q].reshape(-1, 1) \* pure\_x

transform\_x = np.hstack((

transform\_x, tmp\_x[:, q:]

))

return transform\_x

if \_\_name\_\_ == '\_\_main\_\_':

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

x\_train = Transform(x\_train)

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

x\_test = Transform(x\_test)

Eout\_ls = []

for lamb in [0.0001, 0.01, 1, 100, 10000]:

C = 1 / (2 \* lamb)

prob = problem(y\_train, x\_train)

param = parameter('-s 0 -c {} -e 0.000001'.format(C))

m = train(prob, param)

p\_label, p\_acc, p\_val = predict(y\_test, x\_test, m)

Eout\_ls.append(np.mean(y\_test != p\_label))

print(Eout\_ls)

**17. A**

注意只有改第16題的主程式。  
最後結果是[0.09, 0.1, 0.13, 0.195, 0.535]，選Ein(wλ)Ein(wλ)最低的A

from liblinearutil import \*

import numpy as np

import numpy.linalg as la

import random

import math

FILENAME\_TRAIN = "../../hw4\_train.dat.txt" # input file.

FILENAME\_TEST = "../../hw4\_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):

"""

Feature transform x into \Phi(x) = [1, x\_i, x\_ix\_j], for all i != j

"""

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

pure\_x = x[:, 1:]

Q = pure\_x.shape[1]

transform\_x = x.copy()

for q in range(0, Q):

tmp\_x = pure\_x[:, q].reshape(-1, 1) \* pure\_x

transform\_x = np.hstack((

transform\_x, tmp\_x[:, q:]

))

return transform\_x

if \_\_name\_\_ == '\_\_main\_\_':

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

x\_train = Transform(x\_train)

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

x\_test = Transform(x\_test)

Ein\_ls = []

for lamb in [0.0001, 0.01, 1, 100, 10000]:

C = 1 / (2 \* lamb)

prob = problem(y\_train, x\_train)

param = parameter('-s 0 -c {} -e 0.000001'.format(C))

m = train(prob, param)

p\_label, p\_acc, p\_val = predict(y\_train, x\_train, m)

Ein\_ls.append(np.mean(y\_train != p\_label))

print(Ein\_ls)

**18. E**

做出來的EvalEval是[0.2, 0.1375, 0.2375, 0.2625, 0.575]，所以選最好的0.1375。

from liblinearutil import \*

import numpy as np

import numpy.linalg as la

import random

import math

FILENAME\_TRAIN = "../../hw4\_train.dat.txt" # input file.

FILENAME\_TEST = "../../hw4\_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):

"""

Feature transform x into \Phi(x) = [1, x\_i, x\_ix\_j], for all i != j

"""

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

pure\_x = x[:, 1:]

Q = pure\_x.shape[1]

transform\_x = x.copy()

for q in range(0, Q):

tmp\_x = pure\_x[:, q].reshape(-1, 1) \* pure\_x

transform\_x = np.hstack((

transform\_x, tmp\_x[:, q:]

))

return transform\_x

if \_\_name\_\_ == '\_\_main\_\_':

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

x = Transform(x)

x\_train = x[:120, :]

y\_train = y[:120]

x\_test = x[120:, :]

y\_test = y[120:]

Eval\_ls = []

for lamb in [0.0001, 0.01, 1, 100, 10000]:

C = 1 / (2 \* lamb)

prob = problem(y\_train, x\_train)

param = parameter('-s 0 -c {} -e 0.000001'.format(C))

m = train(prob, param)

p\_label, p\_acc, p\_val = predict(y\_test, x\_test, m)

Eval\_ls.append(np.mean(y\_test != p\_label))

print(Eval\_ls)

**19. D**

其實可以直接看第16題的第二個index，即log10λ=−2log10⁡λ=−2的時候的EoutEout，可發現是0.13。

from liblinearutil import \*

import numpy as np

import numpy.linalg as la

import random

import math

FILENAME\_TRAIN = "../../hw4\_train.dat.txt" # input file.

FILENAME\_TEST = "../../hw4\_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):

"""

Feature transform x into \Phi(x) = [1, x\_i, x\_ix\_j], for all i != j

"""

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

pure\_x = x[:, 1:]

Q = pure\_x.shape[1]

transform\_x = x.copy()

for q in range(0, Q):

tmp\_x = pure\_x[:, q].reshape(-1, 1) \* pure\_x

transform\_x = np.hstack((

transform\_x, tmp\_x[:, q:]

))

return transform\_x

if \_\_name\_\_ == '\_\_main\_\_':

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

x\_train = Transform(x\_train)

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

x\_test = Transform(x\_test)

Eout\_ls = []

for lamb in [0.01]:

C = 1 / (2 \* lamb)

prob = problem(y\_train, x\_train)

param = parameter('-s 0 -c {} -e 0.000001'.format(C))

m = train(prob, param)

p\_label, p\_acc, p\_val = predict(y\_test, x\_test, m)

Eout\_ls.append(np.mean(y\_test != p\_label))

print(Eout\_ls)

**20. C**

最後做出來是[0.145, 0.12, 0.15500000000000003, 0.18, 0.5199999999999999]，所以選裡面最小的，也是log10λ=−2log10⁡λ=−2的值：0.12。

from liblinearutil import \*

import numpy as np

import numpy.linalg as la

import random

import math

FILENAME\_TRAIN = "../../hw4\_train.dat.txt" # input file.

FILENAME\_TEST = "../../hw4\_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):

"""

Feature transform x into \Phi(x) = [1, x\_i, x\_ix\_j], for all i != j

"""

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

pure\_x = x[:, 1:]

Q = pure\_x.shape[1]

transform\_x = x.copy()

for q in range(0, Q):

tmp\_x = pure\_x[:, q].reshape(-1, 1) \* pure\_x

transform\_x = np.hstack((

transform\_x, tmp\_x[:, q:]

))

return transform\_x

if \_\_name\_\_ == '\_\_main\_\_':

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

x = Transform(x)

x\_ls = [x[i: i+40, :] for i in range(0, 200, 40)]

y\_ls = [y[i: i+40] for i in range(0, 200, 40)]

Ecv\_ls = []

for lamb in [0.0001, 0.01, 1, 100, 10000]:

C = 1 / (2 \* lamb)

cur\_Eval\_ls = []

for i in range(5):

x\_test = x\_ls[i]

y\_test = y\_ls[i]

x\_train = np.vstack([x\_ls[j] for j in range(5) if i != j])

y\_train = np.hstack([y\_ls[j] for j in range(5) if i != j])

prob = problem(y\_train, x\_train)

param = parameter('-s 0 -c {} -e 0.000001'.format(C))

m = train(prob, param)

p\_label, p\_acc, p\_val = predict(y\_test, x\_test, m)

cur\_Eval\_ls.append(np.mean(y\_test != p\_label))

Ecv\_ls.append(np.mean(cur\_Eval\_ls))

print(Ecv\_ls)

發表於 [**HackMD**](https://hackmd.io/)

 41

[讚賞](https://hackmd.io/@Kaiserouo/Bk8KU6mcD) [收藏](https://hackmd.io/@Kaiserouo/Bk8KU6mcD) [訂閱](https://hackmd.io/@Kaiserouo/Bk8KU6mcD)