Machine Learning HW4

R04922034 吳軒衡

1.

In general, using H' will increase the deterministic noise.

The definition of deterministic noise is the difference between the best h in H and the target function f.

For a given f, since H' is included in H, h' , the best of H' is also in H. Thus if the h' is indeed the best hypothesis for f, it will also be chosen from H. On the other hand, if h of H is the best approximation to the function f,it might not be included in H'.

2.

H(10,0,3) ^ H(10,0,4) = H2
H(10,0,3) => wt =0 for all t >=3
H(10,0,4) => wt =0 for all t >=4
For any HQ to be in H(10,0,3) ^ H(10,0,4) , it must be in H(10,0,3)
=> a polynomial with w0, w1 ,w2 left => H2

3.

Use t for transpose ,x i for iteration count
wi+1 = wi - eta * gradient(Eaug)
= wi - eta (gradient(Ein) + lambda*((wi)t*(wi))/N)
= wi -eta * gradient(Ein) - 2 * lambda * wi / N
= (1 - 2*lambda*eta/N) * wi - eta*gradient(Ein)

=> alpha = (1-2*lambda*eta/N);
beta = (-eta)

4.

wreg = optimal solution to minimize Eaug
=> delta(Eaug) = 0;
wreg = inv(Z'Z+lambda*I) Z'y;
suppose lambda1 > lambda2;
Eaug(wreg1) = Ein(wreg1) + lambda1/ N * wreg1'* wreg1
Eaug(wreg2) = Ein(wreg2) + lambda2/ N * wreg2'* wreg2
(Z'Z+lambda1*I) wreg1 = Z'y;
(Z'Z+lambda2*I) wreg2 = Z'y;
wreg1 = inv(Z'Z+lambda1*I) Z'y;
wreg2 = inv(Z'Z+lambda2*I) Z'y;
Eaug(w) = Ein(w) + lambda / N * w'*w
where lambda acts as a penalty to those w with higher norm(w,2)

Suppose w1 = wreg for labmda1 : Eaug(w1) = Ein(w1) + lambda1 / N * w1'w1
delta Eaug(w) = 0
gradient(Ein(w1)) = -2*lambda1 / N * w1

when use a a smaller lambda = lambda2

Eaug(w1) = Ein(w1) + lambda2 / N * w1'w1

we can find w2 by the above procedure
w2 = (1 - 2*lambda2*eta/N) * w1 - eta*gradient(Ein(w1))
= (1 - 2*lambda2*eta/N) * w1 + eta * 2/N * lambda1*w1'*w1
= (1+ 2*eta(lambda1-lambda2)/N) * w1
Since eta , and labmda1 > lambda2
1+2 * eta(labmda1-lambda2)/ N > 1
=>   norm(w2) > norm(w1)
so   lambda1 > labmda2 => norm(w1) < norm(w2)
=> norm(wreg) is a non-increasing function of lambda for lambda >=0

5.
a (-1,0) b(lo,1) c(1,0)
For h0(x) = b0
leave a out => h0 = 1/2 error = 1/4
leave b out => h0 = 0 , error = 1
leave c out => h0 = 1/2, error = 1/4
For h1(x) = a1x+b1
leave a out , h1(x) : y = (x-1) / (lo-1) error = (h1(-1) - 0)^2 = 4 / (lo-1)^2
leave b out , h1(x) : y =0 , error = 1
leave a out , h1(x) : y = (x+1) / (lo+1) error = (h1(1) - 0)^2 = 4 / (lo+1)^2
1+1/4+1/4 = 1 + 4 / (lo-1)^2 + 4/ (lo+1)^2
1/8 = 1/(lo-1)^2 + 1/(lo+1)^2
1/8 (lo-1)^2 (lo+1)^2 = (lo-1)^2+(lo+1)^2
lo^4 - 2 *lo^2 + 1 = 8 (lo^2-2*lo + 1) + 8 (lo^2 + 2*lo + 1)
lo^4 - 18 * lo^2 - 15 = 0;
lo^2 = 9 + 1/2 (324+ 60)
= 9 +- sqrt(96)
= 9 +- 4*sqrt(6)
lo = sqrt(9+4*sqrt(6))

6.
For at least one person to get all 5 predictions right prior to nth game:
nth-game   :   5 4 3 2 1
number of letters   :   2 4 8 16 32
2 + 4 + 8 + 16 + 32 = 63 -1 = 62
=> 62 letters should be sent before the fifth game

7.
For the fraud to succeed, the minimum letters needs to be sent is the number of letters need to have one person receives correct predictions on all 5 games + the one to ask for money = 63 (Note that we don't need to actually send the prediction of the sixth game)
=> The money can be made by on succeed fraud is 1000 - 63 * 10 = 370

8.

Since we don't know the data, we get one hypothesis by mathematical derivation, thus the size of the hypothesis set is 1.

9.

With M = 1 and finite bin Hoeffding's equation,

E[bad] <= 2 * M * exp(- 2 * eps * eps * N)

= 2 * 1 * exp (-2 * 0.01 * 0.01 * 10000)

= 2 * exp(-2)

= 0.2707

10.

When we get the 0.01 bound on the badness of our hypothesis, what we actually get is a guarantee on the behavior of the customers that are previously approved by the function a(x),

and we have no idea about the potential customers that are previously rejected by a(x).

However, if we apply a(x)*g(x) , we might be able to get rid off those unwanted customers previously approved by a(x).

11.

Use Z to denote Xtilda , u to denote y tilta

minimize E = (Xw-y)' (Xw-y) + (Zw-u)'(Zw-u)

delta (E) = 2/N * (X'Xw - X'y + Z'Zw - Z'u) = 0

(X'X+Z'Z)w = (X'y+Z'u)

w = inv(X'X+Z'Z) (X'y+Z'u)

w = inv(X'X + Xtilda'Xtilda) (X'y+ Xtilda'ytilta)

12.

wreg = inv (X'X+ labmda * I) (X') y

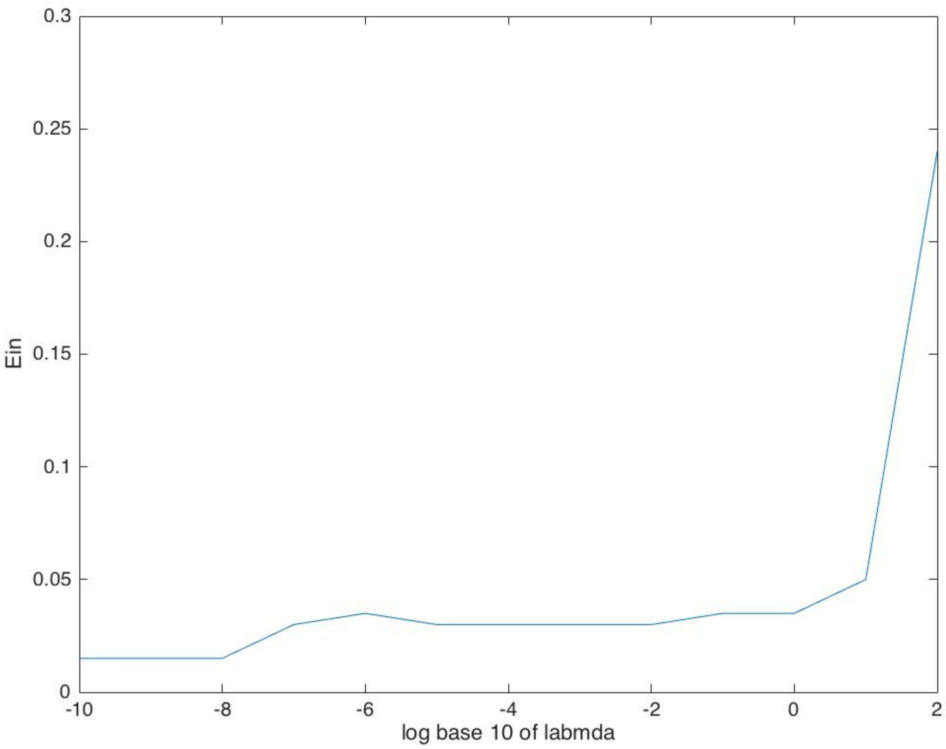
=> ytilta = 0 , Xtilda'Xtilda = labmda I

=> Xtilda = sqrt(labmda) * I

13.

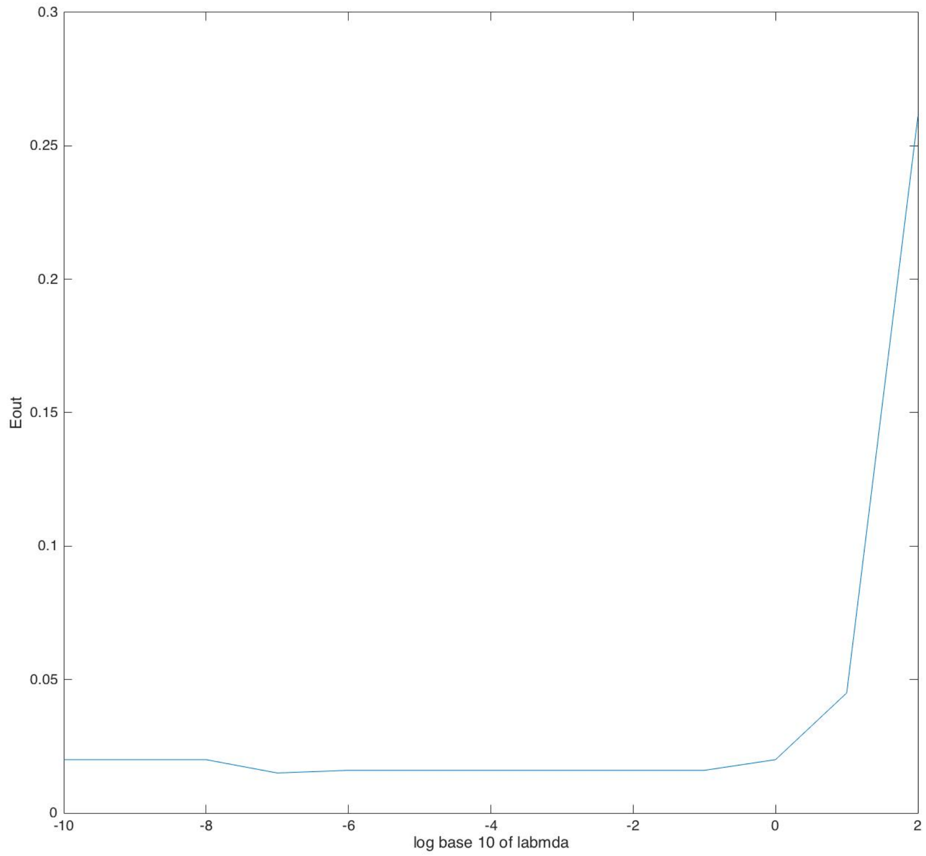
Ein = 5.500000e-02 , Eout =5.200000e-02

14.



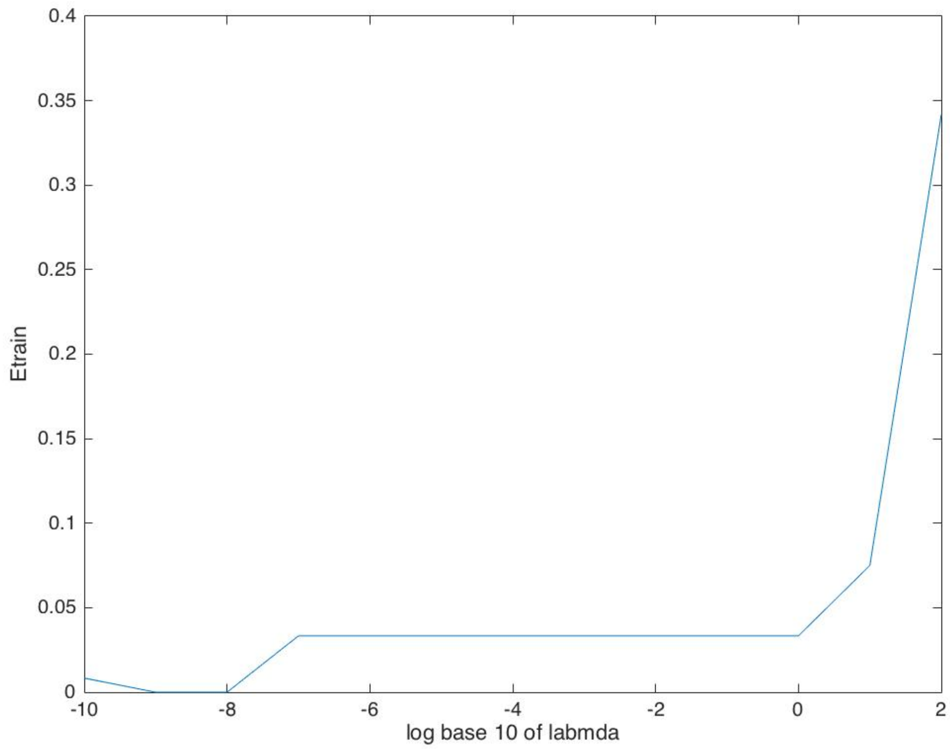
lambda with Min Ein = 1.000000e-08 , Eout = 2.000000e-02

15.



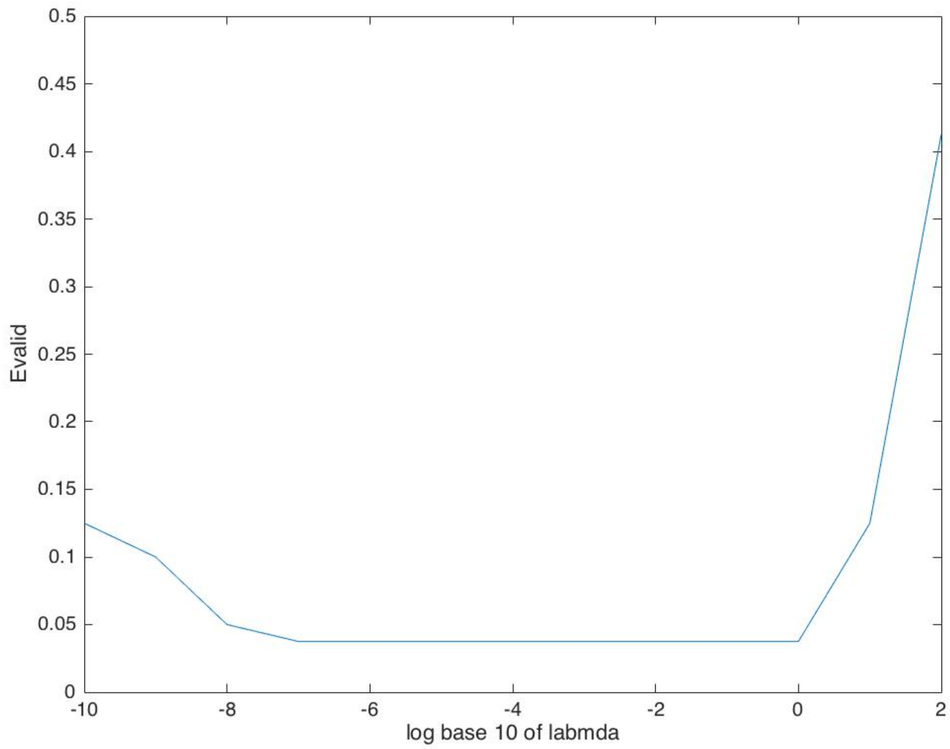
lambda with Min Eout = 1.000000e-07 , Eout = 1.500000e-02

16.



lambda with Min Etrain = 1.000000e-08 , Eout = 2.500000e-02

17.

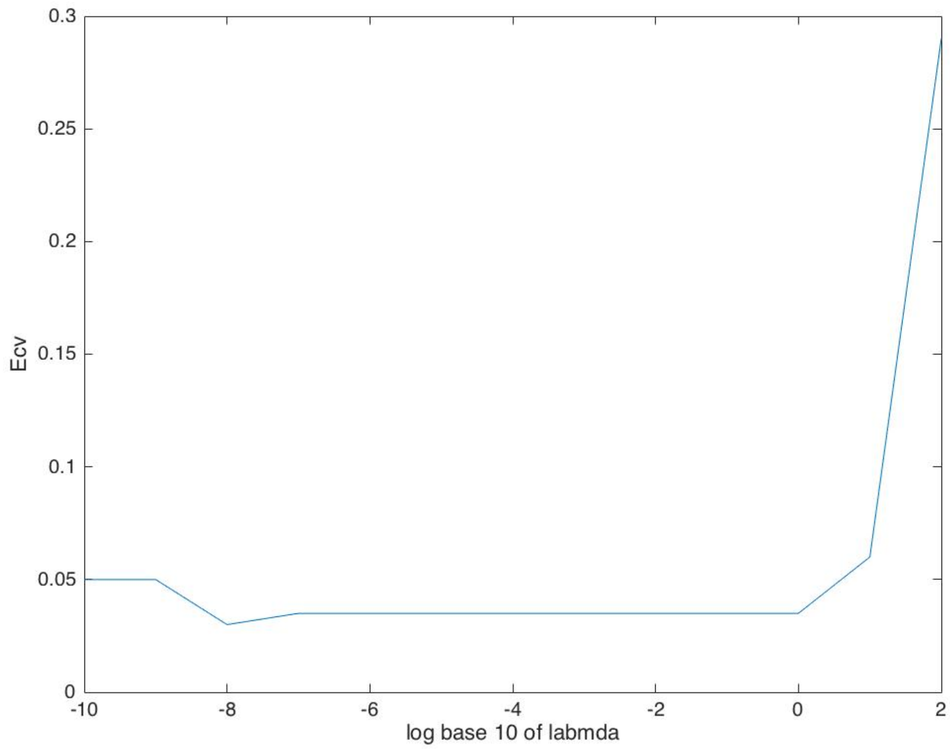


lambda with Min Evalid = 1 , Eout = 2.800000e-02

18.

Ein = 3.500000e-02 , Eout = 2.000000e-02

19.



lambda with min Ecv = 1.000000e-08 , Ecv = 3.750000e-02

20.

Ein = 1.500000e-02 , Eout = 2.000000e-02

21.

use G to denote gamma
Eaug(w) = Ein(w) + lambda/N * w' G'* G * w
delta Eaug = delta (Ein) + 2 / N lambda * (G'Gw) =0 ;
X'Xw - X'y + lambda *(G'Gw)=0;
(X'X + lambda * G'G)w = X'y
wreg = inv(X'X+ lambda* G'G) * X'y
from 11
w = inv(X'X + Xtilda'Xtilda) (X'y+ Xtilda'ytilta)
we can see that ytilta = 0 ;
and Xtilda = sqrt(lambda) *G
or Xtilda = sqrt (lambda) * gamma

22.

use wh to denote whint
Eaug(w) = Ein(w) + lambda/N * (w-wh)'* (w-wh)
delta Eaug = delta (Ein) + 2 / N lambda * (w-wh) =0 ;
X'Xw - X'y + lambda * (w-wh) =0;
(X'X + lambda* I) w = (X'y + wh)
wreg = inv(X'X+lambda*I) *(X'y+wh)

from 11
w = inv(X'X + Xtilda'Xtilda) (X'y+ Xtilda'ytilta)
we can see that
Xtilda = sqrt(lambda)I
wh = Xtilda'ytilta
  = sqrt(labmda)*ytilta
=> ytilta = 1/sqrt(lambda) * wh